8-2013

A Simple, Practical Prioritization Scheme for a Job Shop Processing Multiple Job Types

Shuping Zhang
szhang12@utk.edu

To the Graduate Council:

I am submitting herewith a dissertation written by Shuping Zhang entitled "A Simple, Practical Prioritization Scheme for a Job Shop Processing Multiple Job Types." I have examined the final electronic copy of this dissertation for form and content and recommend that it be accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy, with a major in Management Science.

Mandyam M. Srinivasan, Melissa Bowers, Major Professor

We have read this dissertation and recommend its acceptance:

Ken Gilbert, Theodore P. Stank

Accepted for the Council:
Dixie L. Thompson

Vice Provost and Dean of the Graduate School

(Original signatures are on file with official student records.)

A Simple, Practical Prioritization Scheme for a Job Shop Processing Multiple Job Types

A Dissertation Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Shuping Zhang

August 2013

DEDICATION

To my parents, Duo and Xuebing,

my wife, Yi,

and my son, Xinchen

for their love, encouragement, and numerous sacrifices

ACKNOWLEDGEMENTS

At the end of all these intense years I would like to thank many people who helped me through this PhD journey. I first would like to thank my PhD advisors and dissertation committee chairs, Dr. Mandyam Srinivasan and Dr. Melissa Bowers, for supporting and guiding me throughout the years. Dr. Srinivasan introduced me into the fields of Management Science and the Theory of Constraints. He played a major role in forming my understanding of the fields and he also devoted so much of his time and effort to my dissertation work and taught me patiently about the meaning of research and scholarship. Dr. Bowers offered her comprehensive knowledge and strategic insights into my dissertation. She always encouraged me as a researcher to try new things and experiment. She has taught me much more about Operations Research from technical skills to philosophical ideas.

I am very grateful to the remaining members of my dissertation committee, Dr. Kenneth Gilbert and Dr. Theodore Stank, for serving on my committee and making numerous suggestions to improve my final manuscript.

Very special thanks go to Ms. Jane Moser who warmly welcomed me into the program as a program coordinator. She looked after me so well all these years. I am always grateful for her kindness and support.

It is difficult for me to find the right words to express my gratitude to my family members. I am deeply thankful to my parents who raised me with a love of science and supported me in all my pursuits. I am also deeply thankful to my wife Yi and my son Xinchen for their love, support, and sacrifices. Without them, this dissertation would never have been written.

ABSTRACT

The maintenance, repair, and overhaul (MRO) process is used to recondition equipment in the railroad, off-shore drilling, aircraft, and shipping industries. In the typical MRO process, the equipment is disassembled into component parts and these parts are routed to back-shops for repair. Repaired parts are returned for reassembling the equipment. Scheduling the back-shop for smooth flow often requires prioritizing the repair of component parts from different original assemblies at different machines. To enable such prioritization, we model the back-shop as a multi-class queueing network with a ConWIP execution system and introduce a new priority scheme to maximize the system performance. In this scheme, we identify the bottleneck machine based on overall workload and classify machines into two categories: the bottleneck machine and the non-bottleneck machine(s). Assemblies with the lowest cycle time receive the highest priority on the bottleneck machine and the lowest priority on non-bottleneck machine(s). Our experimental results show that this priority scheme increases the system performance by lowering the average cycle times without adversely impacting the total throughput.

The contribution of this thesis consists primarily of three parts. First, we develop a simple priority scheme for multi-class, multi-server, ConWIP queueing systems with the disassembly/reassembly feature so that schedulers for a job-shop environment would be able to know which part should be given priority, in what order and where. Next, we provide an exact analytical solution to a two-class, two-server closed queueing model with mixed non-preemptive priority scheme. The queueing network model we study has not been analyzed in the literature, and there are no existing models that address the underlying problem of deciding prioritization by job types to maximize the system performance. Finally, we explore conditions under which the non-preemptive priority discipline can be approximated by a preemptive priority discipline.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Introduction and Motivation

The research presented in this dissertation was motivated by a real-world problem arising in a maintenance, repair, and overhaul (MRO) environment. The MRO process is used to recondition equipment in a variety of industries such as the railroad, off-shore drilling, aircraft, and shipping industries. In the typical MRO process, the equipment is disassembled into component parts and these parts then are routed to back-shops for repair. Repaired parts are returned for reassembling the equipment. Scheduling the back-shop for smooth flow often requires prioritizing the repair of component parts from different original assemblies at different machines. To this end, the back-shops can be modeled as job-shops.

A typical job-shop is a high-mix, low-volume (HMLV) production system that simultaneously processes a diverse mix of jobs using shared resources. These jobs typically have different routings, due dates, priorities, quantities, and material and resource requirements. The essence of scheduling such job-shops is to determine how to allocate scarce resources, in the form of machines times, to perform a collection of activities, known as jobs [E. L. Lawler (1993)]. One of the earliest known analytical studies in job-shop scheduling is undertaken by Muth and Thompson [Muth and Thompson (1963)].

To model the complex diversity and uncertainty inherent in most real-world job-shops, a number of simplifying assumptions are often made. For instance, when the job-shop is modeled as a network of queues, a simplifying assumption is that each resource can only process one job at

most at any point in time. This thesis models the back-shop as a multi-class queueing network with a ConWIP (*con*stant *w*ork *i*n *p*rocess) execution system [Hopp (2008)], and we introduce a new priority scheme to maximize the system performance.

1.2 Models for Manufacturing Systems

Schmidt and Taylor [Schmidt and Taylor (1970)] define a *system* as a collection of entities, *e.g*., workers, machines, customers, which interact together to accomplish some logical end. In practice, the term *system* usually refers to the actual facility or process being analyzed and the specific meaning of a *system* depends on the particular study. Our study is concerned with manufacturing systems, more specifically the flow of defective parts through the various resources (machines) in the system that processes those parts.

Conceptually, a system is often viewed as a black box which processes an input signal to generate an output signal. The *state* of a system is defined as a collection of variables necessary to describe a system at a particular time, on a particular object [Law (2006)]. Based on the characteristics of the state, systems can be categorized into two types: *discrete* and *continuous*. A *discrete* system is a system with a countable number of states, *i.e*., the state variables can change at discrete points in time. An automobile manufacturing facility, a shopping center, and a bank are examples of discrete systems. A *continuous* system is one for which the state variables change continuously over time. An industrial plant that produces chemicals is an example of continuous system.

There are a variety of ways manufacturing systems can be modeled to evaluate their performance or to get insights into the relationships among various components of the system. See Figure 1.1.

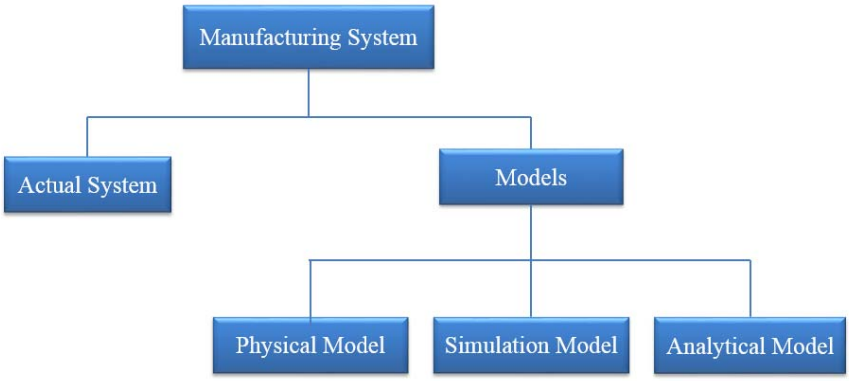


Figure 1.1 Manufacturing System Representations

The ideal way to study a system is to study the actual system itself, varying the parameters of the system physically and letting it operate under new conditions. However, often this is either not possible or too expensive. Therefore, we develop models to predict the performance of the manufacturing system. A *model* is a representation of the operations of the actual system, *i.e.*, a simplified view of the system that facilitates the analysis of the system. Essentially modeling is the art of selective simplification. Models shield our brains from numerous trivial and distracting details and allow us to concentrate on the fundamental processes in manufacturing systems. Three types of models are commonly used to analyze manufacturing systems: physical models, simulation models, and analytical models [J. A. Buzacott (1993)].

Physical models (also called *iconic* models) are miniatures or prototypes of the real system. The major difference from the real system is that the physical model uses a different dimensional scale. Physical models can use toy size components, but may be provided with a control system that employs the same logic as the real system. They are excellent tools to train workers and gain insights into the actual system. However, it can be very hard for a physical model to capture the long-run behavior of the system or certain statistical properties, such as machine downtime.

Simulation models represent the events that could occur during system operations through a sequence of steps in a computer program, allowing the logical relationships that exist between events to be described in detail. The probabilistic nature of many events, such as machine failure, can be represented by sampling from a distribution that describes the pattern of the event occurrence. Thus to represent the typical behavior of the system it is necessary to run the simulation model for a sufficiently long time, so that all possible events can occur a reasonable number of times. Simulation models are often used along with an interactive graphic display to demonstrate the movement of jobs and material handling devices. This can be of great value in communicating the assumptions of the model to manufacturing engineers and others.

Analytical models describe the system using mathematical or symbolic relationships. These models are used to derive a formula, to define an algorithm, or to develop computational procedures so that the performance measures of the system can be calculated. Analytical models can also be used to demonstrate properties of various operating rules and control mechanisms. Sometimes it is not possible to obtain the performance measures from the relationships describing the system within a reasonable amount of computer time or space. In this case, we have to make further assumptions which can modify these relationships. The resulting model is then approximate rather than exact. Since testing the approximation may require a simulation model, approximate models are only useful if they are easy to implement and can provide insights into what determines the system behavior.

Considering the nature of this thesis, we restrict our analysis to simulation and analytical models, addressed in Chapter 3 and Chapter 4 respectively.

1.3 Applications of Queueing Models in Manufacturing Systems

The simplest manufacturing system is a single machine, worker, or facility. The machine (worker or facility) has to perform a task, or a set of tasks, or jobs, or customers. Throughout this thesis, we assume that jobs/customers are discrete entities. Each job will require a processing time (service time) that may vary between successive repetitions of identical jobs. The actual time required is known before processing, but sometimes the time cannot be predicted with any accuracy in advance because it will also depend on problems encountered while processing the job.

Rooted in the studies of A. K. Erlang [Erlang (1917)] on telephone networks and in the creation of Markov processes by A. A. Markov [Norris (1998)], queueing theory is widely used to model manufacturing systems as well as a variety of other real-world systems. Queueing theory studies the conflicts between unpredictable arrivals and finite-capacity resources. The essence of queueing theory is to study the effects of such arrivals on system congestion.

The simplest queueing model involves a single queue (Figure 1.2), in which jobs (customers) enter from the left and exit at the right. The circle represents the "server". For instance, in a repair facility the server would be a machine handling the defective parts. The open rectangle in Figure 1.2 represents the waiting line, or queue, that builds up ahead of the server.

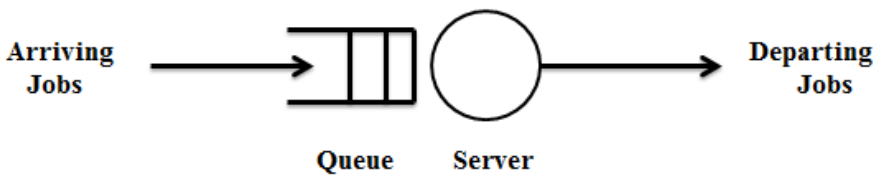


Figure 1.2 A Single Server Queueing System

In the single server queueing system, the server can only serve one customer at a time and hence it is either in a "*busy*" or an "*idle*" state. If the server is busy when the customer arrives, the newly arriving customer waits for service, assuming that there is enough waiting space. When the customer currently in service departs, one of the waiting customers is selected for service according to a queueing (or scheduling) discipline.

Kendall [Kendall (1953)] introduces a shorthand notation, known as *Kendall's notation*, to describe elementary queueing systems:

$$A/B/m,$$

where A identifies the distribution of the inter-arrival times; B identifies the distribution of the service times; and m is the number of servers $(m \geq 1)$ in the system. For example, *M/M/1* is used to describe the *Single-Server Exponential Queueing System*, in which customers arrive at a single-server service station in accordance with a Poisson process having arrival rate λ (the M identifies a Markovian arrival/service process). That is, the times between successive arrivals are independent exponential random variables having mean $1/\lambda$. Each customer, upon arrival, goes directly into service if the server is free or waits in line if the server is busy. When the server finishes serving a customer, the customer leaves the system, and the next customer in line, if there is any, enters service. The successive service times are assumed to be independent exponential random variables having mean $1/\mu$.

When manufacturing systems are a bit more involved, the queueing model representing the actual system becomes more complex. Even the analysis of a single work station with arbitrary inter-arrival and processing times (*G/G/1*) is quite complicated and relies on complex-variable techniques or random walk models. Analysis for such models can be found in Wolff (1989),

Kleinrock (1975), and Gross and Harris (1974). In general, even when these models are analyzed assuming Markov processes, the state-space of the queueing model explodes. For example, consider a station with *s* machines. This system can be modeled as a *G/G/s*-queue. To describe this simple queueing system as a Markov process, one has to keep track of the states of all servers over time. The analysis becomes even more complex when queueing networks are used to model the system. A queueing network is a collection of interconnected queues representing service centers (system resources) subject to different types of customer demands. To obtain the exact results, most of the literature on queueing theory incorporates extra simplifying assumptions, such as the assumption of Poisson arrivals or exponentially distributed processing times. In practice, the inter-arrival and processing times are usually far from exponential. There is also much literature available on approximations for more complicated queueing systems, but most approximations apply only to specific systems, or are not significantly accurate.

1.4 Literature Review of Multi-class, Multiple-Server Closed Queueing Networks

Multi-class, multi-server queueing networks are widely applied in computer systems, modern communication networks, road transportation networks, and parallel manufacturing systems. In a multi-class, multi-server closed queueing network, each class of customers will have its own parameter configuration (including routing and service time distributions). They are commonly used to model service systems with many servers. Randhawa and Kumar (2009), Maglaras and Zeevi (2003, 2004, 2005) provide different applications for these models.

In a parallel server system, customers are handled by a set of server pools and leave the system after service. Similar to multi-class queueing networks, the exact analysis of parallel server

systems is limited to a few special cases. Even when available, the results from the exact analysis provide limited insight on the general properties of performances of these systems and can rarely be used for optimization purposes.

A natural question to ask is: why are we using multiple classes? First of all, it is natural to categorize the network into various classes based on their own performance measures. For example, in a flexible manufacturing system, if a machine produces three different types of parts, it is important to measure the in-process inventory of each of them individually. Then, it makes sense to model the system using three different classes (job types). Second, when the service times are significantly different for different types of customers, it might be beneficial to classify the customers based on their service times. For example, in most grocery stores, there are special checkout lines for customers who have fewer items. Third, due to physical reasons of where the customers arrive and wait, it might be practical to classify customers. For example, at fast-food restaurants customers can be classified as drive-through and in-store depending on where they arrive.

A queueing network in which jobs do not enter or leave the system is called *closed* queueing network. The closed queueing network allows the analyst to model systems where the number of jobs (work-in-process) in the system is constant. In a multi-class queueing network, some of the classes can be closed while other job classes can be *open* in the sense that customers can enter or depart from the system. If a queueing network contains both open and closed classes, it is called a *mixed* queueing network. Inside the queueing network, job classes can differ in configurations, *i.e*., their service time distribution, and in routing probabilities. These models also allow a job to change its class when it transfers from one server to another.

The next question is: when there are multiple classes, how should the customers be served? That is, we need to determine a service discipline to serve the customers. Typical considerations affecting service disciplines include cost, fairness, performance, physical constraints, goodwill, and customer satisfaction. Different service disciplines and their applications in queueing networks will be discussed in Chapter 4.

Multi-class, multi-server closed queuing network models provide a convenient framework to evaluate the impact of population constraints on the stochastic interactions between different classes at various servers of the network. Solutions to these queuing networks quantify the impact of these network interactions on performance measures such as throughput, queue length, and waiting times at individual servers of the system.

Generally, multi-class, multi-server closed queuing networks are usually categorized into two types: product-form (PF) networks and non-product form networks. The solution to the queueing network is called a *product-form solution* if the solution for the steady-state probabilities can be expressed as a product of factors describing the states of each node. The product-form solution for queueing networks is introduced in Jackson (1963), which considers open queueing networks with exponential service time distributions at each station. The analysis in Jackson (1957) and Jackson (1963) are regarded as a breakthrough in the analysis of queueing networks.Gordon and Gordon (1967) extends Jackson's results to derive product-form results for single closed queuing networks. Later on, the classic Baskett, Chandy, Muntz, and Palacios (1975) extends these results to open, closed and mixed queueing networks with multiple classes, generally distributed service times, and different queueing disciplines. This model is also well known in literature as the *BCMP network*. Solutions to product-form networks are often obtained using either the

convolution algorithm [Buzen (1980)] or the Mean Value Analysis algorithm [Reiser and Lavenberg (1980)]. However, a big limitation of product-form queuing networks is that the assumptions that guarantee product-form solutions are often restrictive and not valid in many practical applications.

In fact, in most practical settings, the assumptions required for product-form solutions are not satisfied. Consequently, alternative solution algorithms have been proposed for non-product-form closed queuing networks. Among these techniques, two techniques of interest include the aggregation technique and Marie's method [Marie (1979)]. The main idea behind both approaches is to replace the original queueing network with an equivalent product-form network. The only difference between these two approaches are how they obtain the parameters of the equivalent product-form network. When the size and complexity of the queueing network grow, the number of arrival rates and service rates for different servers increases significantly. Consequently, product-form solutions for large closed queueing networks are prohibitively costly. Ryan, Baynat, and Choobineh (2000) propose an alternative approach based on the solution of a non-linear programing problem. The formulation they use bridges the connections between network throughput and population constraints. Although quite attractive as a possible alternative approach to obtain product-form approximations, this method does not explicitly capture the different types of variability present in multi-class closed queuing networks.

1.5 Objective and Contribution

The ultimate goal is to improve the system performance by reducing the average cycle time, namely the time a customer spends in the system, and increasing the total average throughput.

The contribution of this thesis consists primarily of three parts. First, we develop a simple priority scheme for multi-class, multi-server, ConWIP queueing systems with the disassembly/reassembly feature so that schedulers for a job-shop environment would be able to know which part should be given priority, in what order and where. Next, we provide an exact analytical solution to a two-class, two-server closed queueing model with mixed non-preemptive priority scheme. The queueing network model we study has not been analyzed in the literature, and there are no existing models that address the underlying problem of deciding prioritization by job types to maximize the system performance. Finally, we explore conditions under which non-preemptive priority solutions can be approximated by preemptive solutions.

1.6 Structure of the Thesis

The first chapter introduces the core problem and the modeling techniques for manufacturing systems, and then translates these techniques into a queueing system perspective. It also reviews the literature on multi-class, multiple-server closed queueing networks, which form the theoretical foundation for this research. At the end of chapter 1, we briefly address our contribution to the body of knowledge.

Chapter 2 starts with the motivation of our research, and then provides detailed descriptions of the problem and the literature review for job-shop modeling.

Chapter 3 handles the simulation models by using the priority scheme we develop. We start with the priority scheme and then give a literature review of the bottleneck identification, which is a

key factor in our model. Modeling with *ExtendSim* is introduced, and the average total throughput and average total cycle time are compared for different models and different scenarios. We first define a baseline model, which uses the first-come-first-served (FCFS) service discipline, and then a priority model, which applies the head-of-the-line (HOL) service discipline. Different numerical examples/results are provided.

Chapter 4 deals with the analytical models for mixed priority queueing networks. We first give an overview of the priority queues and the literature for mixed priority systems. We focus on two types of mixed priority queueing network: preemptive priority and non-preemptive priority. The analytical model discussed here is a two-class, two-server closed queueing network with preemptive/non-preemptive priorities at each server. We provide the exact analytical solutions to each model. Also, we explore conditions under which non-preemptive solutions can be approximated by preemptive solutions.

Chapter 5 summarizes this thesis and gives directions for future research.

CHAPTER 2

BACKGROUND

This chapter presents the back-shop scheduling problem with the goal of minimizing the mean overall flow time and/or maximizing the mean overall throughput. The chapter is organized as follows. Section 2.1 briefly introduces the motivation of this research work. Section 2.2 describes the back-shop scheduling problem and its essential features, and then introduces the concept of rule-based priority schemes. Section 2.3 provides a literature review of job-shop modeling.

2.1 Motivation

The job-shop scheduling problem is a classical problem that many manufacturing organizations face in their daily operations. One of the most important aspects of the problem is resource contention: there are limited resources (machines) and the capacities of these resources (machines) are finite. Thus, when considering the routing and product mix problems in a job-shop, the flows among different resources cannot be assumed to be independent. To model resource contention explicitly when scheduling complex job-shops, it is necessary to identify the system bottleneck first, which allows prediction of the workflows and flow times based on this knowledge.

The performance of a job-shop is influenced considerably by the sequence in which different types of jobs are processed at different machines. The job-shop problem is studied in the context of the maintenance, repair, and overhaul (MRO) process for engines in the aircraft industry. This research provides a simple approach for setting priorities in job-shops to improve system mean response times without sacrificing overall mean throughput.

2.2 Problem Description and Essential Features

2.2.1 Problem Description

Overview of the System

In the typical MRO process for aircraft engines, the engines are first disassembled into component parts. These component parts are then tested separately. If found defective, the parts are routed to a back-shop for repair. Otherwise they are sent to a holding area. Repaired parts are returned to this holding area for reassembly. When all parts are ready in the holding area, they are reassembled and the completed engine is delivered to the customer (see Figure 2.1).

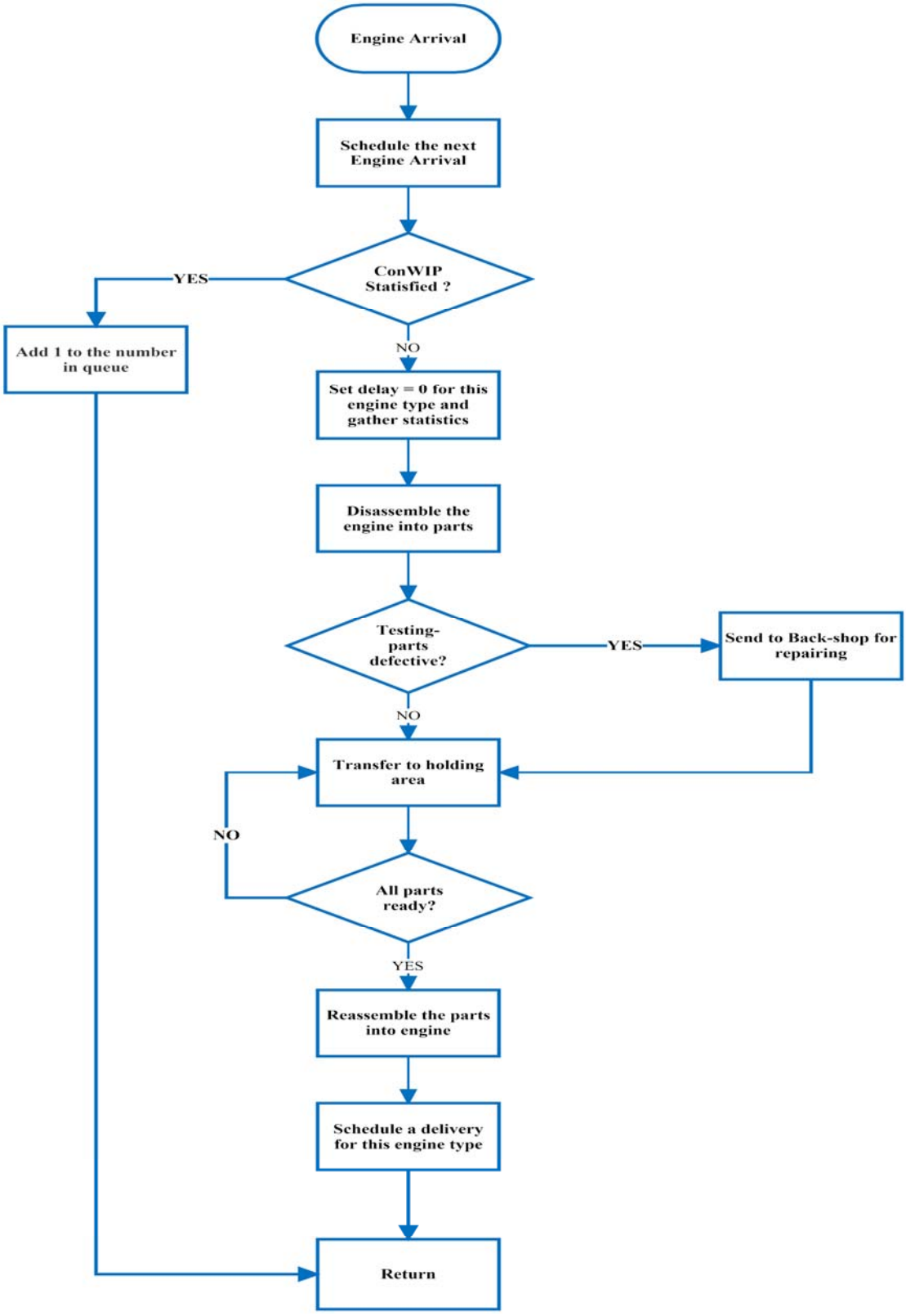


Figure 2.1 Flow of the System for One Engine Type

The Back-Shop Flow

All the defective engine parts are sent to the back-shop for repair. Figure 2.2 presents a typical back-shop configuration used in this thesis. This configuration consists of six machines labeled $M_1, M_2, \ldots, M_6$. It is assumed that there are three possible engine types repaired in this facility. Each engine type will be disassembled into three parts: Part A, Part B, and Part C. Thus, there are nine "engine part-types" and each engine part-type follows a given routing. For instance, engine type 1 part *A* is processed at machine M_1, then at machine M_3, M_4, and M_6. Engine type 2 part *C* is processed at machine M_1, then at machine M_2, returning back to M_1 before finally being processed at M_4. Similarly, engine type 3 part *B* passes through machine M_3, M_5, and M_6. Note that different engine parts may require different processing time on each machine.

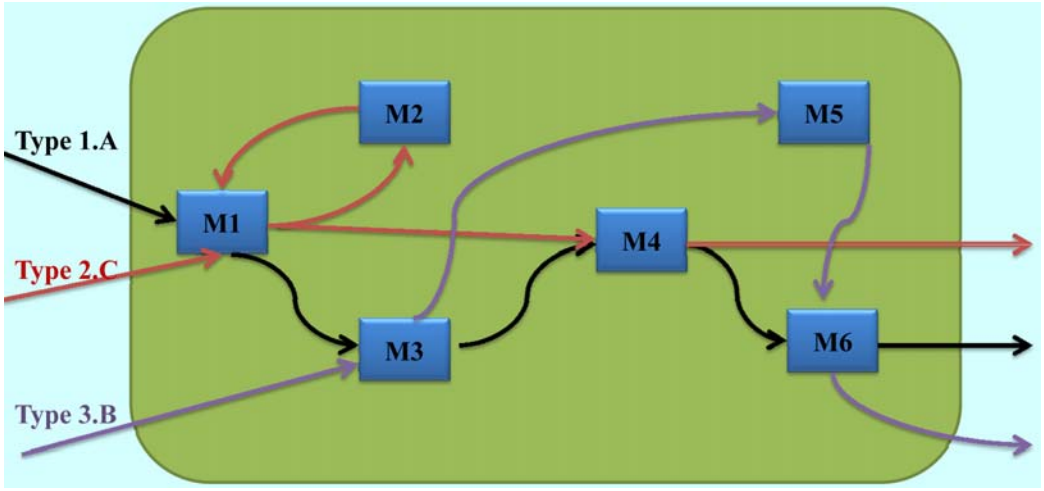


Figure 2.2 Simplified Layout of the Back-shop Flow

Figure 2.2 also shows that there are instances where resources (machines) need to be shared. Under these circumstances, scheduling the back-shop for smooth flow often requires prioritizing the repair of component parts from different original assemblies at different machines.

2.2.2 Essential Features of the Problem

This section discusses the essential features of the model used in this research. Four aspects of the problem are studied here: ConWIP protocol, disassembly/reassembly process, priority scheme, and multiple job classes.

ConWIP Protocol

The *con*stant *w*ork *i*n *p*rocess (ConWIP) mechanism is used to model a job-shop in which a new job is introduced into the system every time a job completes its service and departs from the system. The ConWIP mechanism presents a straightforward way to establish a cap on the *w*ork *i*n *p*rocess (WIP) in a production line.

From a modeling perspective, a ConWIP system belongs to a special class of closed queueing networks, in which customers (jobs) never leave the system, but instead circulate around the network indefinitely. Of course, in practice, the arriving jobs are different from the departing jobs.

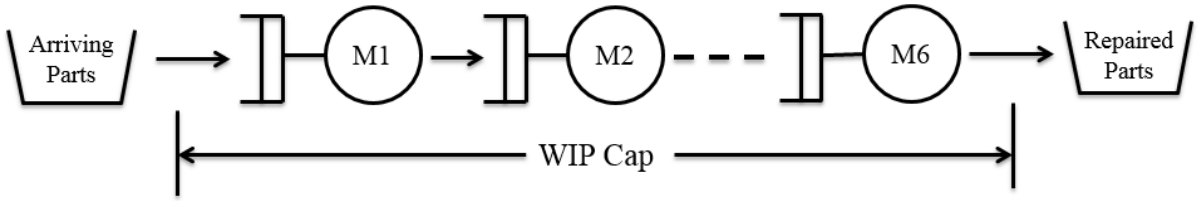


Figure 2.3 A ConWIP System

Disassembly / Reassembly Process

Disassembly/Reassembly (or *fork-join*) systems are broadly applied in many manufacturing companies and MRO facilities for electronics, automobiles, and aircrafts, in which a job is disassembled into several parts, and each part will be processed in parallel at different machines.

Upon the completion of the process, these parts will be reassembled into one product and delivered to the customer.

From the modeling perspective, Disassembly/Reassembly systems belong to a class of parallel processing networks, in which synchronization is required [Bolch, Greiner, de Meer, and Trivedi (2006)]. The parallel programs (jobs) consist of a series of tasks that have to be processed in a certain sequence (order). The structure of a disassembly/reassembly system is shown in Figure 2.4. The fork-join system disassembles an arriving job (for instance, an aircraft engine) into *3* tasks (parts) that arrive simultaneously at the *3* parallel processes *Process 1, Process 2,* and *Process 3*. Task 1 and 2 will join *Process 4* once they are processed. Similarly, task 3 will join *Process 5*. In the fork-join system, as soon as a job is served, it enters the join queue and waits until all tasks are done. As depicted in Figure 2.4, in the end, all tasks will merge into one job and leave the system.

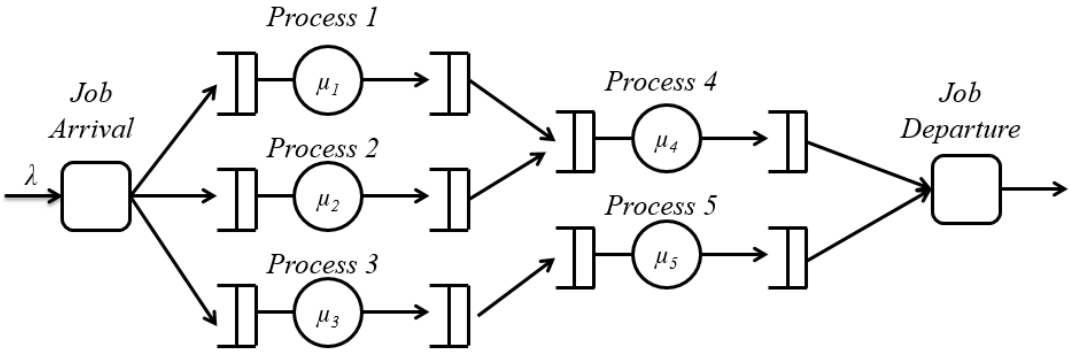


Figure 2.4 *An Elementary Disassembly/Reassembly (Fork-Join) System*

Previous research on Disassembly/Reassembly queueing systems focused mainly on the effects of the uncertainty of the processing times at the reassembly stages and subassembly stages. Among these studies, Hemachandra and Eedupuganti (2003) considers a fork-join queue for an

open system. Krishnamurthy, Suri, and Vernon (2004) treats a fork-join queue as a closed system. Approximation methods are developed in their studies. An exact analysis of an assembly system is presented by Gold (1998). In next chapter, we will provide a more general and robust framework to deal with the multi-class, multi-parts fork-join queueing network.

Priority Scheme

Priorities are often applied in queueing networks as a mechanism for service or resource allocation. Selecting appropriate rules of priorities can greatly improve system performances, such as improvements on throughput, cycle time, utilization, and the balance of flow. For a single queue, there is only one class of job that arrives at the service facility. Each job in the class is assumed to be identical in all respects except service time requirement. In practice, different jobs in a manufacturing system are usually not treated in the same manner. Instead, they are distinguished according to some "measure of importance," and therefore priorities are used to indicate this relative measure of importance [Jaiswal (1968)].

This research work will focus on a simple rule-based priority scheduling scheme, which is a heuristic approach for the job-shop scheduling problem. The rationale behind this rule-based priority scheme is twofold: first, the priority enables balancing the flows by granting higher priorities to slow-running job at non-bottleneck machines; second, it avoids overloading the bottleneck machine by giving the fast-running jobs the higher priority at the bottleneck machine. By protecting the most valuable resource — the bottleneck machine, this scheme increases the performance of the whole system in terms of higher mean throughput and lower mean cycle time.

Multiple Job Classes

In practice, job-shops or MRO facilities are fed with multiple job types. In our aircraft engine repair model, we assume that multiple engine types will arrive independently for repair, and each engine type can be disassembled into several parts. Different engine part types pass through different routings in the back-shop, and may require different service times even at the same machine. What complicates the model is that because of the resource contention (machine-sharing), the inter-arrival time between different classes is no longer independent; instead, they are highly correlated. These correlations make the computation of large scale analytical models economically infeasible. State space size under ConWIP and priority settings explodes. For example, in Chapter 4, we will discover that even for a very small product mix, the exact solution for the analytical model is difficult to obtain when the non-preemptive priority discipline is considered. For our case, since the analytical model becomes extremely difficult or even impossible to characterize, the simulation method becomes a natural and ideal tool for use in modeling job-shops. The simulation model will be presented in Chapter 3.

2.3 Job-shop Modeling and Literature Review

Garey and Johnson (1979) stated the Job-shop Scheduling Problem (JSSP) as one of the hardest combinatorial optimization problems, with JSSP being amongst the worst members of the class of NP-hard problems. Since the late 60's, a significant amount of literature about JSSP has appeared in the field of operations research and management science. Also, it has been the object of intense research from different perspectives. Many formulations and solution methods have been proposed. In this review, we only focus on modeling and algorithm development using queueing networks. More comprehensive bibliographies and literature reviews on job-shop scheduling can be found in the research conducted by Sisson [Sisson (1959, 1961)] and

Thompson [Giffler and Thompson (1960)]. More recent reviews are attributed to Fan and Ren-qian (2010) and Jain and Meeran (1999).

A Brief Description of the Job-shop

A job-shop consists of several different types of equipment (machines), and each equipment type performs a specific set of production operations, for instance, drilling, milling, or forming. In any job-shop, a customer (job) passes through a sequence of machines as specified in its routing configuration and the customer may wait for the required resources at those machines. Job-shop scheduling models usually make the following assumptions [Gere (1966) and J. A. Buzacott (1993)]:

1. Each machine can process no more than one operation at a time, *i.e.*, the capacity of the machine is 1.
2. Each machine operates independently of other machines in the shop.
3. Machines are available all the time, and there is no break down.
4. Each operation, once initiated, must be performed to completion (no preemptive priorities).
5. Each job, once accepted, must be performed to completion, *i.e.*, no cancellation or interruption is allowed.
6. Each job requires a finite processing time to perform each operation. Each operation must be completed before any operation which it must precede can begin (no "lap-phasing"). The operation time includes set up time.
7. The time intervals for processing are independent of the order in which the operations are performed. (In particular, set up times are sequence-independent and transportation time between machines is negligible.)
8. Work-In-Process inventory is allowed.
9. Due dates are known and fixed.
10. The job routing is given and no alternative routings or dynamic routings are permitted.

Job-shop vs. Flow-shop

The various shop scheduling problems puts restrictions on how the operations can be scheduled on the various machines. Two typical shop scheduling problems are flow-shops and job-shops.

In a typical *flow-shop* setting, arriving jobs are processed in order through a specific sequence of machines: machine1, machine2,…, concluding with the last machine. The arrangement of the flow is a linear structure (Figure 2.5), although in some cases, a job can skip some machine(s), for example, the job routing could be: machine 1→machine 3→ machine4.

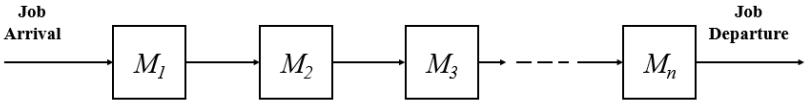


Figure 2.5 Flow-Shop

A job-shop is different from a flow-shop. In a *job-shop* environment, jobs can be processed on machines in any order, that is, the flow of jobs is not unidirectional, which is the major distinguishing feature of a job-shop. Simply put, in a job-shop, customers can be introduced to the shop at almost any machine, and the routing of a job may require it to return to a given machine several times (see Figure 2.2).

Models and Algorithms

Researchers have developed various solution methods to handle job-shop modeling problem. Roughly, the techniques used can be divided into two categories: optimization approaches and approximation methods. The classical job-shop scheduling problem, where the objective is to minimize the makespan (the time between the start and the finish of a sequence of jobs or tasks), is an NP-hard combinatorial problem. The algorithms used are able to provide optimal or near optimal solutions to many practical problems. However, as the problem size increases, the computational cost associated increases exponentially. To solve this dilemma, various approximation methods are developed to tackle the scheduling problem. A classification of

algorithms for job-shop modeling is depicted in Figure 2.6. More detailed and complete classification of the job-shop scheduling techniques is discussed in Jain and Meeran (1999).

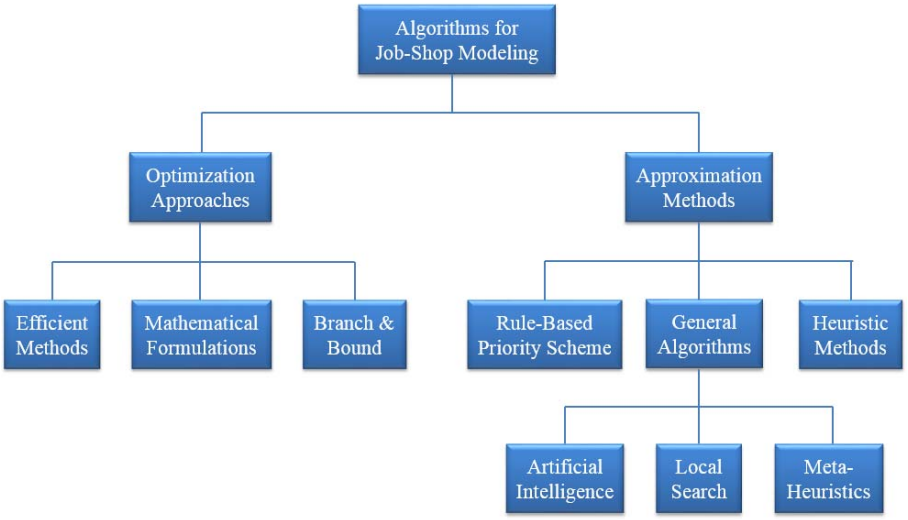


Figure 2.6 Algorithms for Job-Shop Modeling

Theoretically, a job-shop can usually be viewed as a multi-class open queueing network. Jackson first develops a queueing network model of a dynamic job-shop, which assumes a Poisson external arrival process, exponentially distributed service times, and Markovian job transfers between machines [Jackson (1957, 1963)]. Inspired by Jackson's work, Suri and Walrand [Suri (1983), Walrand (1988)] introduce the ideas of quasi-reversibility, local balance, and station balance for job-shop modeling. Later on, Kühn (1976) extends Jackson's model to systems with general service times using FCFS service protocol. Buzacott and Shanthikumar (1985) extends Kühn's approach to job-shops with service time-dependent disciplines. This decomposition approach employed by Kühn, Shanthikumar, and Buzacott is refined by Whitt (1983). Approximations for the second moment of the sojourn time are reported in Buzacott and Shanthikumar (1985), and the extension to approximate its distribution and the application to due

date setting is described in J. George Shanthikumar (1988). Further refinement and extensions for multi-class job-shop systems are reported in G. R. Bitran and Tirupati (1988). More comprehensive reviews of the job-shop modeling problem are referred to G. Bitran and Dasu (1992), Sultana Parveen (2010), Jain and Meeran (1999), and Veronique Sels (2012).

Although rule-based priority schemes usually cannot outperform the exact solution obtained from optimization procedures, in practice, they are widely used because of the easy implementation, the low computational complexity, and the model robustness. Therefore, for practical reasons (size of the problem, complexity of the scheduling environment, or lack of scheduling software), simple and easy-to-understand priority rules are welcomed by practitioners. Finally, due to the dynamic nature of the job arrival process, simple priority rules can easily determine what type of jobs get higher priority during the manufacturing process to improve the system performance. The early work on rule-based priority schemes is done by Jackson [Jackson (1954, 1957)], Giffler and Thompson [Giffler and Thompson (1960)], and Gere [Gere (1966)], among which, Giffler and Thompson (1960) is viewed as the common basis of all rule-based priority schemes. A series research efforts conducted by Conway are considered the most comprehensive experimental studies, which deal with the elementary processing and due date attributes-oriented rules with sophisticated refinements [Haupt (1989)].

CHAPTER 3

A CYCLE TIME BASED PRIORITY SCHEME

In this chapter, we will develop a new priority scheme and present numerical results from the empirically study. First, we review the bottleneck identification problem in literature and explain how this relates to our model. Then, we introduce a new cycle time based priority scheme for a class of multi-class, multi-server priority models. Third, we briefly introduce *ExtendSim*, the simulation software used for the experiment. Finally, we conduct a large number of experiments to demonstrate that our priority scheme meets the research objective stated in Section 1.5, namely, improving the system performance by lowering average total cycle times without adversely impacting the average total throughput. At the end of this chapter, we will summarize the experimental results and draw conclusions.

3.1 Notation and Definitions

3.1.1 Notation

Note: We will refer to Engine Type j as Class j customer or Type j job in all of our discussions henceforth. The network under discussion is a multi-class closed queueing network, denoted by

μ_{mj}, mean service rate of Engine Type j at machine m.

μ_{mji}, mean service rate of Engine Type j Part i at machine m.

V_{mj}, visit ratio (or relative throughput) of Engine Type j at machine m.

V_{mji}, visit ratio (or relative throughput) of Engine Type j Part i at machine m.

S_{mj}, mean service time of Engine Type *j* at machine m.

S_{mji}, mean service time of Engine Type *j* Part i at machine m.

$WL_{mj} = V_{mj} \times S_{mj}$, the mean workload (or service demand) of machine *m* for class-*j* customers.

$WL_{mji} = V_{mji} \times S_{mji}$, the mean workload (or service demand) of machine *m* for Engine Type *j* Part i.

$I = \{A, B, C\}$, the set of Engine parts indices, $i \in I$.

$J = \{1,2,3\}$, the set of Engine Types or customer class indices, a member of class-*j* is called a class-*j* customer, $j \in J$.

$M = \{1,2,3, \dots ,6\}$, the set of machine indices, $m \in M$.

N_j, the (constant) number of class-*j* customers (Engine Type *j*) circulating in the network.

N, population vector, **N** = (N_1, N_2, N_3), the total product mix.

$N = N_1 + N_2 + N_3$, the total number of customers in the system.

$P_{ji,}$ the probability that Engine Type *j*, Part *i* needs to be sent to back-shop for repair.

TH_j, the throughput of class-*j* customer.

CT (*m, j*), mean cycle time within machine *m* for a Type *j* job including the time spent in the queue plus the time spent processing.

WIP, mean (time-averaged) work-in-process (Def. 3.3).

We may introduce more notation as needed in our later discussions.

3.1.2 Definitions

Definition 3.1 *Cycle Time* (also referred to as *average cycle time* or *flow time*): is the time that a customer spends within a system, *i.e.*, the time from job release to job completion. Cycle time is routing dependent. *Cycle time* and *flow time* will be used interchangeably in this thesis.

Definition 3.2 *Throughput* (or *throughput rate*): refers to the number of jobs per unit time leaving the system after completing processing. It is the average output of an operation process (machines, line) per unit time (*e.g.*, parts per hour).

Definition 3.3 *Work-In-Process*: is the number of customers in the system including customers waiting in the queue and customers undergoing processing.

Definition 3.4 *Visit Ratio* (or *relative throughput*): is the average number of visits per production cycle to a particular machine.

Definition 3.5 *Workload* (or *service demand*): is a set of all inputs that the system/machine receives from the arriving process during a given period of time. The relationship among workload, visit ratio, and service time is:

$$WL_{mji} = V_{mji} \times S_{mji},$$

where

WL_{mji}— the workload of Engine Type-*j*, Part *i* on machine *m*.

Also, we can calculate the total workload on machine *m*

$$WL_m = \sum_j \sum_i WL_{mji}.$$

Definition 3.6 *Routing*: is the sequence of machines through which a job is processed in the facility.

Definition 3.7 *Configuration*: is defined by the routing of a class of customers together with the mean service time for each machine that these customers visit.

3.2 Bottleneck Identification

3.2.1 Literature Review

When the performance or capacity of an entire system is limited by a single or limited number of components or resources, there is a *bottleneck*. Goldratt and Fox [Goldratt, Cox, and Whitford (2004); Goldratt and Fox (1986)] redefine the concepts of *bottleneck* and *non-bottleneck* in the environment of a modern manufacturing system. A *bottleneck* is any resource whose capacity (the available time for production) is less than or equal to the demand placed on it. A *non-bottleneck* is a resource whose available capacity is greater than the demand placed on it. Goldratt and Fox point out that *bottleneck* and *non-bottleneck* resources are defined in a general sense and include tools, fixtures, operators, machines, setup personnel, engineers, maintenance teams, etc. Based on these definitions, they infer that bottlenecks govern both the throughput and the inventory of the system, and therefore, to optimize the system one should try to balance the flow, not the capacity. Goldratt's classification and definition have a profound impact on how people understand the way that modern manufacturing systems work.

The bottleneck analysis arises from the computational complexity of obtaining an exact solution when modeling computer systems and networks. As the number of classes, customers, jobs, machines, and stations increases, the computational cost becomes prohibitively expensive and more time consuming. For a separable open queueing network, Jackson's Theorem can be used

to compute the joint distribution, and for a closed queueing network, Mean-Value Analysis (MVA) can be used to compute the mean performance measures. For non-separable queueing networks, there are no general techniques available. Even so, we can derive the system performance metrics by analyzing the bottleneck queues, or get an approximate solution by decomposing the network. The exact analysis of such queueing network becomes impossible when priority rules are used.

An excellent and complete survey of bottleneck analysis can be found in Schweitzer [Paul J. Schweitzer (1993)]. The survey considers both exact bottleneck analysis for product-form closed queueing networks and approximations for non-product-form closed queueing networks. They remarked that different methods for bottleneck analysis will be fundamental for future system modeling; approximation and asymptotic bottleneck analysis techniques will play more important roles.

Reiser (1979) provides a method for heuristically identifying the bottleneck of a multi-class queueing network by devising an iterative algorithm for solving product-form queueing networks with large customer populations. He formulates a closed multi-chain queueing model for a communication network with end-to-end window flow control on virtual channels. The solution technique is a heuristic extension of the mean-value analysis of Reiser and Lavenberg (1980). Their results are reported to be asymptotically valid for large population sizes.

Pittel (1979), Lavenberg (1980) , and Anselmi (2008) study the asymptotic analysis for identifying bottlenecks of queueing networks through the linear programming approach. Pittle considers networks in which certain stations have capacity constraints and derives the asymptotic expressions for the queue length distributions. Lavenberg derives asymptotic throughput rates

under the assumption of the existence of infinite server (delay) stations, and under heavy load conditions these stations act as Poisson sources, making the rest of the network behave as an open model. Anselmi et al. introduce a new framework supporting the bottleneck analysis of closed, multi-class BCMP queueing networks where a *BCMP* network is a class of queueing networks for which a product-form equilibrium distribution exists [Robertazzi (1990)] in the limiting regime where the number of jobs proportionally grows to infinity while other input parameters remain fixed.

Balbo and Serazzi (1996) and Balbo and Serazzi (1997) provide a detailed study on bottleneck analysis for multi-class closed queueing networks using asymptotic analysis. They derive asymptotic expressions for throughputs, utilizations, mean queue lengths, and mean cycle times for multi-class product-form queueing networks with load-independent servers.

Casale and Serazzi (2004) show how the theory of convex polytopes can be applied to the bottleneck analysis of multi-class queueing networks. Their algorithm is used to study large scale models consisting of thousands of servers and several dozens of customer classes, all serving an arbitrary number of customers. The algorithm is also used to understand graphically the behavior of models with up to three distinct customer classes. The experimental results show that additional information concerning the actual bottlenecks of product-form networks can be obtained efficiently if the number of model classes is less than 10.

Note that all these bottleneck analyses in literature rely heavily on asymptotic analysis. The algorithms or methods researchers developed previously are limited to specified types of queueing networks. Most of them focus on product-form closed queueing networks. Although these research provides insight on how we understand and model manufacturing systems, they are still quite involved and not intuitively clear. Therefore, it is very hard to put them into

practical use. In next section, we will provide a heuristic approach on how to identify the bottleneck of a system, which can easily be implemented by practitioners or even schedulers.

3.2.2 A Heuristic Approach

Consider a back-shop facility, which can be modeled as a closed queueing network. Suppose the capacity of machine *m* is 1, with average service time S_{mj} per class *j* job. Let V_{mj} be the average number of visits for class *j* customers to machine *m* per time unit. The bottleneck machine of the system is obtained as follows:

Step1: Calculate the workload of all class-*j* customers on machine *m*

$$WL_{mj} = V_{mj} \times S_{mj}, \forall m, j$$

where *m* = 1,2,…, M, *j* = 1, 2, …, J

Step 2: Identify the bottleneck machine of the system based on maximum workload

$$\mathbf{B} = \arg\max_{m} \max_{j} WL_{mj}$$

The bottleneck machine has the highest utilization among all machines. Relieving a bottleneck by speeding up its servers (*i.e.*, reducing WL_{mj}) or adding servers only shifts the bottleneck to another queue. Identifying a bottleneck in a network is important. Goldratt suggests that the bottleneck determines the performance of the whole system. Time lost at a bottleneck is time lost to the whole system and time saved at a non-bottleneck is just a mirage [Goldratt (1984)].

3.3 The Rule-Based Priority Scheme

Consider a back-shop (MRO) facility. Scheduling the back-shop operations for smooth flow often requires prioritizing the repair of component parts from different original assemblies at different machines. To enable such prioritization, we model the back-shop as a multi-class

queueing network with a ConWIP execution system and introduce a new priority scheme to maximize the system performance. In this scheme, first, we identify the bottleneck machine based on overall mean workload for each machine and classify machines into two categories: *bottleneck* machine and *non-bottleneck* machine(s). Then, we implement a FCFS (First-Come-First-Serve) prioritization system for all parts on every machine in the back-shop. Based on the results obtained from FCFS operations, assemblies with the lowest cycle time receive highest priority on bottleneck machine and lowest priority on non-bottleneck machine(s).

There are five steps to implement this priority scheme:

Step1: Calculate the workload for every customer/job on each machine based on the given configuration.

Step2: Sum up the total workload on each machine.

Step3: Identify the bottleneck machine based on maximum total workload. All other machines are classified as non-bottleneck machines.

Step4: Prioritize all operations in the back-shop using FCFS (First-Come-First-Serve) order, calculate the average cycle time $CT_j\ , \forall j$ for class-*j* customer.

Step5: Priorities will be given based on the value of $CT_j\ , \forall j$. For the bottleneck machine, customers with lower cycle time will be granted higher priorities. The priority order will be reversed for non-bottleneck machines, *i.e*., customers will be processed in order from highest cycle time first to lowest cycle time last.

3.4 A Brief Introduction to Simulation with *ExtendSim*

3.4.1 Introduction

ExtendSim (formerly known as *Extend*) is a powerful, leading edge simulation tool, which can develop dynamic models of real-life processes in a wide variety of fields with different types of system configurations. Each *ExtendSim* product has components aimed at specific market segments, but all products share a core set of features. For our research purpose, *ExtendSim* will be used to enable the building-block style creation of discrete event models that represent the MRO processes. In *ExtendSim*, a model is constructed by selecting blocks from libraries (Items, Value, Plotter, etc.), placing the blocks at appropriate locations in the model window, connecting the blocks to indicate the flow of entities (or values) through the system, and then detailing the blocks using dialog boxes. Blocks are used to model activities, resources, and the routing of jobs throughout the process. Additional blocks are used to collect data, calculate statistics, and display output graphically with frequency charts, histograms, and line plots. The most basic block diagram of a simulation for a business process starts with the representation of the activities in the process; then the conditions that determine the routing of jobs through the process, and if appropriate, a representation of resources, are also included. In addition, specific rules (sometimes, customized rules) are used to define the operation of queues and the utilization of available resources. Finally, data collection and displays of summarized output data are added to the model.

3.4.2 Modeling Process with *ExtendSim* Scenario Manager

It would be tedious to conduct and repeat the experiments by using different configurations of the simulation model. Fortunately, this can be done automatically, at least partly, with the

implementation of the new *Scenario Manager* block in *ExtendSim* 8, which can increase the efficiency of the experiment significantly.

Scenario analysis systemically and strategically examines the outcome of different model configurations. The purpose is to support the exploration and analysis of alternatives and gain insights into why the system behaves the way it does and how it can be improved and managed. *ExtendSim* facilitates scenario analysis through the *Scenario Manager* block (Value library), which can be added to any model to control all aspects of the analysis. The *Scenario Manager* essentially keeps track of multiple what-if scenarios, all based on the same model. It offers a highly flexible framework for experimentation and analysis.

There are seven steps to perform scenario analysis:

Step 1: add a Scenario Manager Block to the model

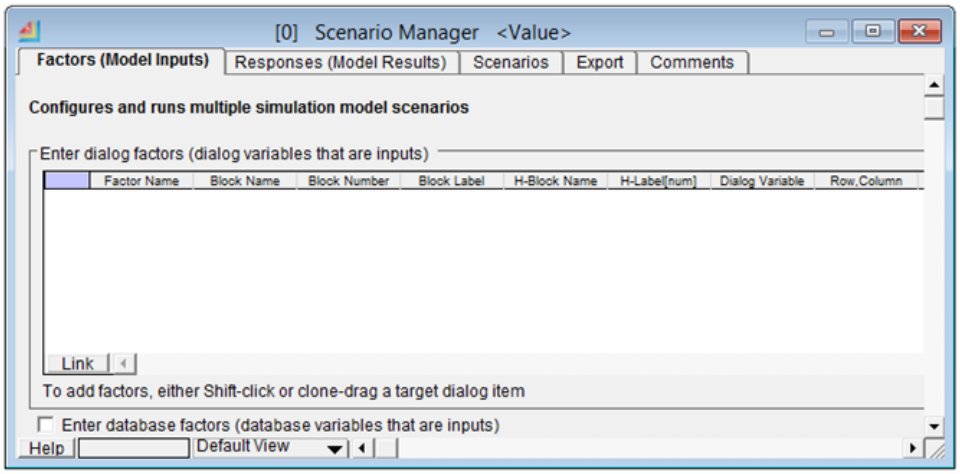


Figure 3.1 Scenario Manager Block

From the above screen shots, we can see that the Scenario Manager has several tabs: Factors (Model Inputs), Responses (Model Results), Scenarios, and Export. Those tabs represent the steps in using the Scenario Manager to run experiments.

Step 2: identify and add factors and values (model inputs) to be included in the experiment

After adding the Scenario Manager Block to the model, the next step is to determine and add model parameters. As shown in Figure 3.2, there are three factors in the simulation model: WIP1 (Work-In-Process for Engine Type 1), WIP2 (Work-In-Process for Engine Type 2), and WIP3 (Work-In-Process for Engine Type 3). The initial value (*Minimum*) for each parameter is 6; the ending value *(Maximum*) is 60; the incremental (*Step*) is 6.

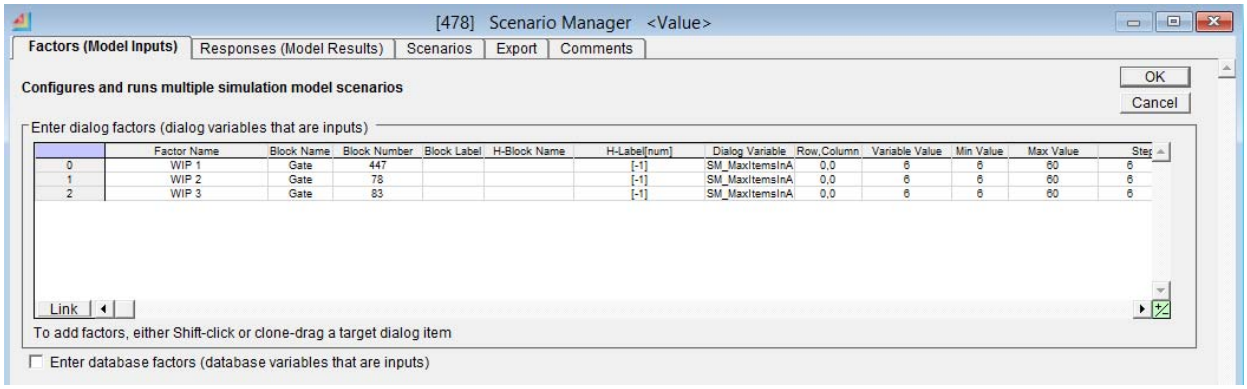


Figure 3.2 Factors (model inputs) of Scenario Manager

Step 3: identify and add the responses (model results) to be analyzed

The next step is to identify and add targeted responses (model results) to the Scenario Manager. The results of interest in this study are average throughput for each engine type (meanTH_1, meanTH_2, and meanTH_3) and average flow time for each engine type (meanFT_1, meanFT_2, and meanFT_3). The screen shot of the Response tab is illustrated below:

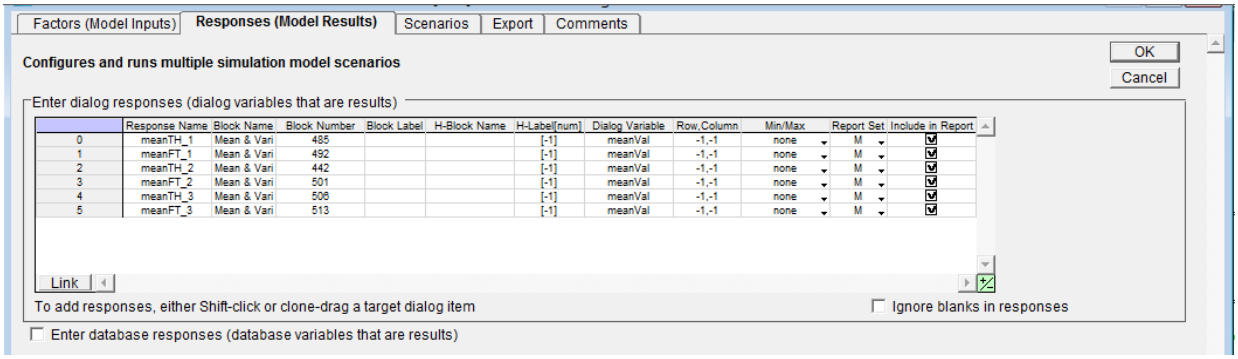


Figure 3.3 Responses (model results) of Scenario Manager

Step 4: determine what you want in the final report

For this study, only the average values of the responses for each scenario are shown in the final report.

Step 5: determine the design of experiments, then generate and run the scenarios

ExtendSim 8 provides three DOE (design of experiments) methods: *Manual design*, *Full factorial design*, and *JMP custom design*. The DOE method in this research is **Full factorial design**.

Experimental *factors* are model inputs that are purposefully changed to study the resulting effects. The values of these factors are *levels*. An experimental run involves a specified level for each factor. A full factorial experiment consists of every combination of the levels of factors in the experiment. Thus, if we have *3* factors, each at ten levels, the full factorial generates all of the combinations of the factor values [Mee (2009)]. There will be a total of 1,000 rows (10×10×10 levels) in the Scenarios table.

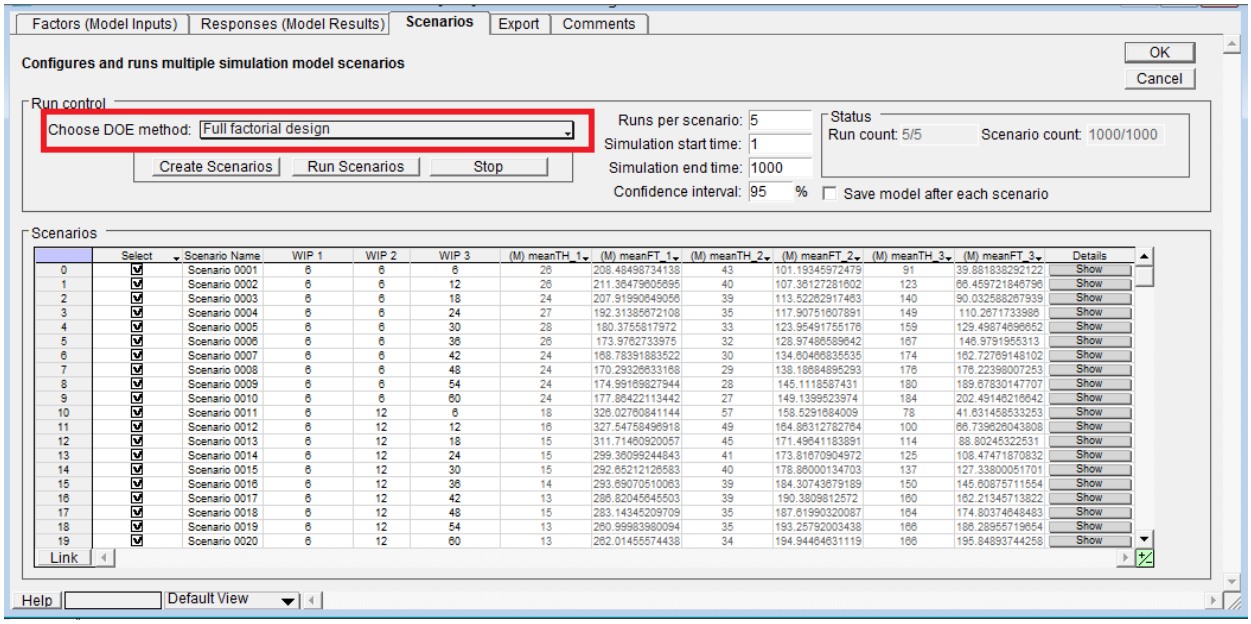


Figure 3.4 DOE of Scenario Manager

Step 6: analyze the results

Detailed analysis will be shown in Section 3.6 Numerical Results.

Step 7: exporting the scenarios to Excel for further analysis

For further analysis, we export the results from each scenario to Excel. The data is combined and sorted. Then, the statistics (total mean values of throughput and flow time) are plotted for comparison of different models.

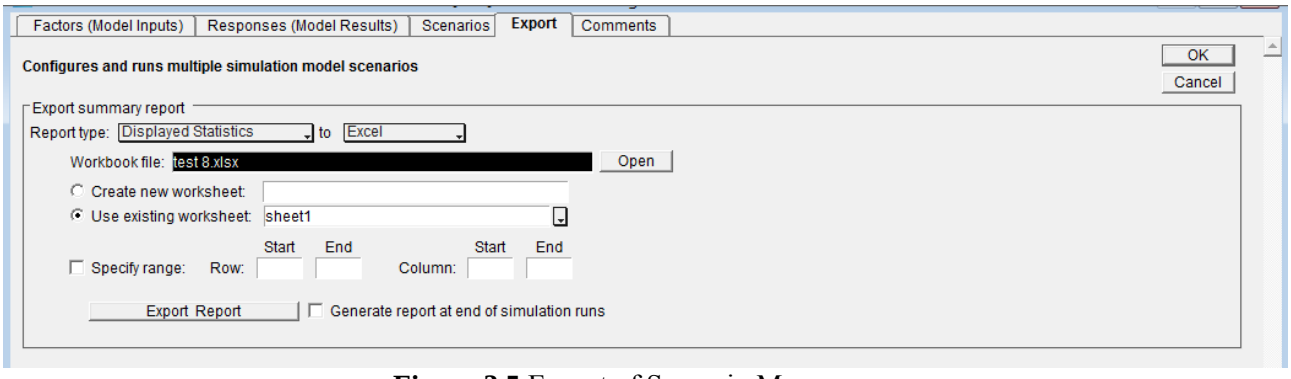


Figure 3.5 Export of Scenario Manager

3.5 General Experimental Framework of the Simulation Models

3.5.1 Brief Description

The operational process in the MRO facility is modeled as a multi-class queueing network operated with the constant work-in-process (ConWIP) protocol. Since the facility processes multiple engine parts, each engine type part is modeled as a separate customer class in the queueing network. First, each Engine Type is disassembled into three component parts and these parts will be tested independently; then defective parts are routed to the back-shop for repair and all other parts will be sent to the holding area; finally, repaired parts are returned for reassembling the equipment. Figure 3.6 illustrates the disassembly, test, repair, and reassembly processes for one engine type using *ExtendSim*.

To enable the prioritization for smooth flow due to resource contention (machine sharing), we implement the new priority scheme developed in Section 3.3 to maximize the system performance.

Figure 3.7 presents the structure and flow of the back-shop using *ExtendSim*.

3.5.2 Experimental Assumptions

Since the research purpose is to study the effectiveness of the priority scheme, we assume:

(1) All service times are exponentially distributed.

(2) The time taken for disassembly, testing, and reassembly is negligible.

(3) Setup time and changeover time between different parts at each machine in the back-shop is also negligible.

(4) The capacity of each machine is 1. For any given time, there is zero or one part at each machine.

(5) Each machine operates independently of other machines in the back-shop.

(6) There are no machine break downs.

(7) Each operation, once initiated, must be performed to completion.

(8) The job routing is given and no alternative routings are permitted.

3.5.3 Basic Settings and Layout of Each Example

Three different types of engines, labeled as Engine Type 1, Engine Type 2, and Engine 3, are considered for the simulation. Each engine type consists of three parts: Part A, Part B, and Part C. Therefore, all together nine different types of parts are analyzed in the experiments.

For each numerical example, first, the configuration table is given. The table includes routings (visit ratios), service time, and failure rate (the probability that the part is defective and needs to be sent to back-shop for repair); then the overall mean workload for each machine is calculated. Based on the calculation, the bottleneck machine is identified.

As discussed in the last section, each example has 1,000 scenarios. For each scenario, the simulation starting time is 1, and the ending time is 10,000. Five runs per scenario (*i.e.,* replication) on each measure of performance (*i.e*., flow time and throughput) are recorded. For each scenario, we calculate the following two system performance metrics:

$$Total\ average\ Flow\ Time = \sum_j CT_j, j = 1, 2, 3.$$

$$Total\ average\ Throughput = \sum_j TH_j, j = 1, 2, 3.$$

where j is the Engine Type indices, CT_j and TH_j are average flow time and average throughput for class-j Engine Type respectively.

For benchmark purpose, two types of models will be used in the experiments:

(1) Baseline model, in which FCFS is applied everywhere.

(2) Non-preemptive priority model, in which the priority rule follows the scheme developed in Section 3.3.

At the end of each numerical example, total average flow time and total average throughput are compared between the baseline model and the priority model for total WIP = 30, 60, 90, 120, and 150.

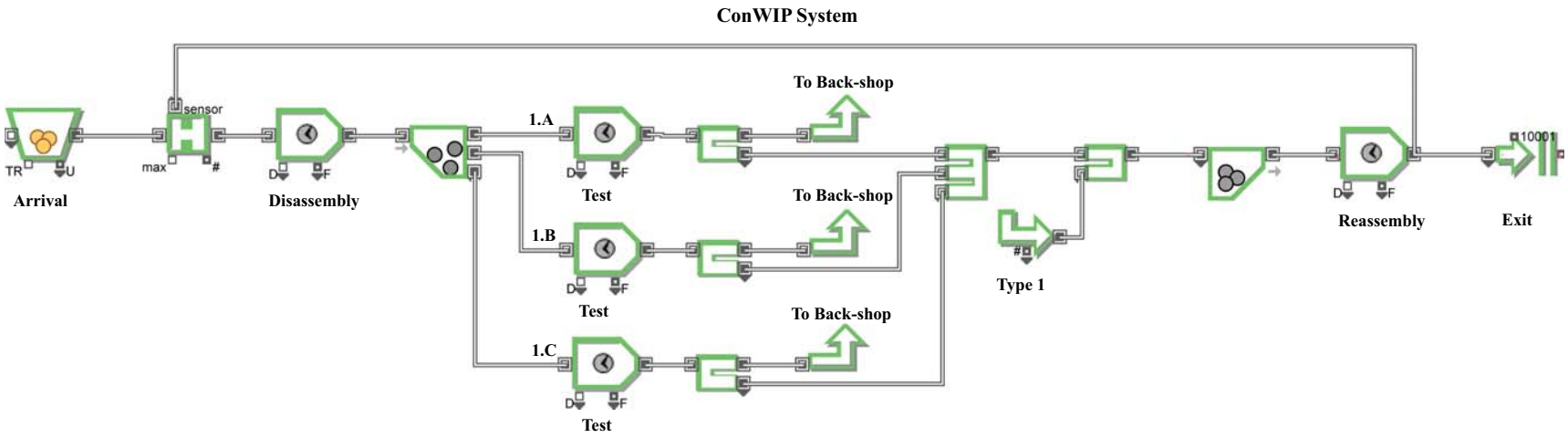


Figure 3.6 Flow of the System using *ExtendSim*

Note: This is the simulation system for Engine Type 1. Engine Type 2 and Engine Type 3 have similar structures. Also, for illustration purpose, this figure has been simplified.

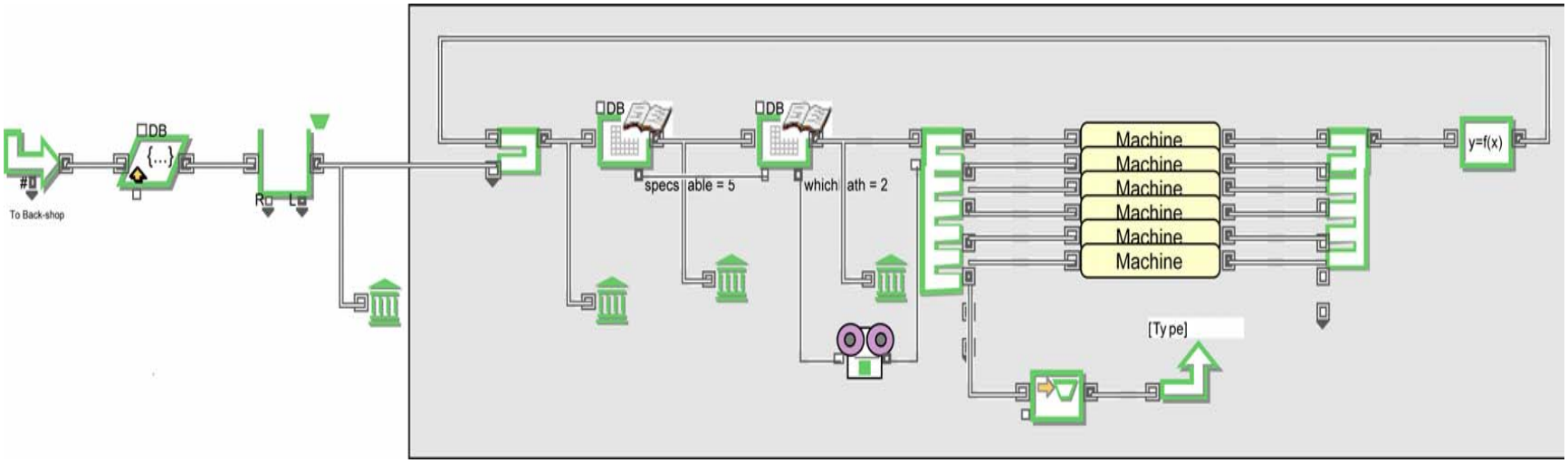


Figure 3.7 Flow of Back-shop Using *ExtendSim*

3.6 Numerical Results

We conducted a number of experiments to investigate the effectiveness of the priority scheme discussed in Section 3.3.

3.6.1 Numerical Example 1

Let's start with a simple example. In this example, we assume that there are three engine types in the back-shop and for each engine type, only one part fails with the probability of 100%. The configuration of this example is:

Table 3.1 Configuration of Numerical Example 1

Machine	V (Visit Ratio)			S (Service Time)			WL (Workload) = V*S			
	Eng. 1	Eng. 2	Eng. 3	Eng. 1	Eng. 2	Eng. 3	Eng. 1	Eng. 2	Eng. 3	Sum
Machine 1	1	1	1	2	2	2	2	2	2	6
Machine 2	3	1		4	10		12	10	0	22
Machine 3	1	1	1	3	13	3	3	13	3	19
Machine 4	1			4			4	0	0	4
Machine 5			1			5	0	0	5	5
Machine 6		1			5		0	5	0	5

Based on our definition, machine 2 has the highest overall workload, therefore, it is the bottleneck machine, *i.e.*, **B** = 2.

The experiment results are illustrated below:

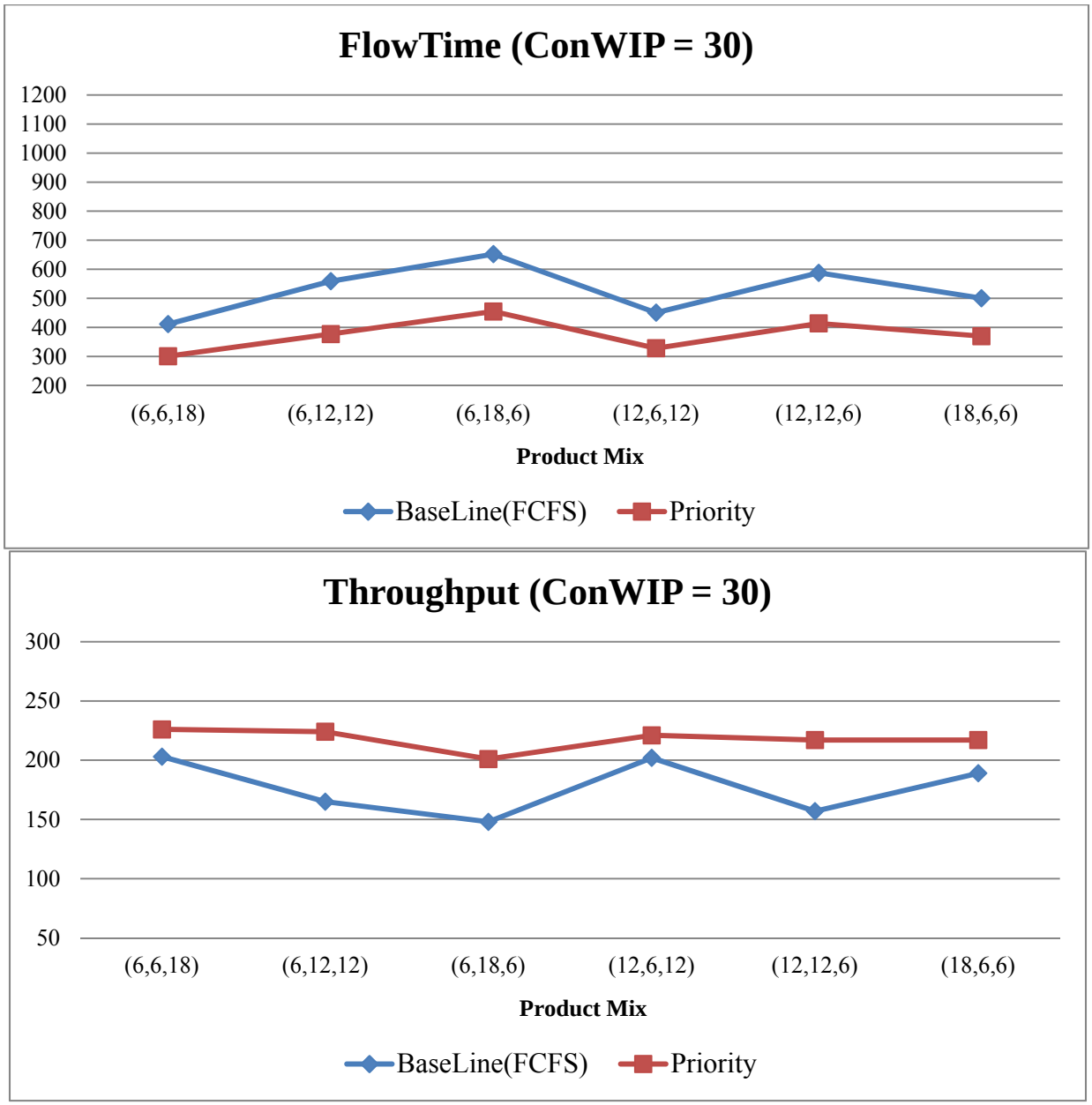


Figure 3.8a Numerical Results 1: Baseline (FCFS) vs. Priority (ConWIP = 30)

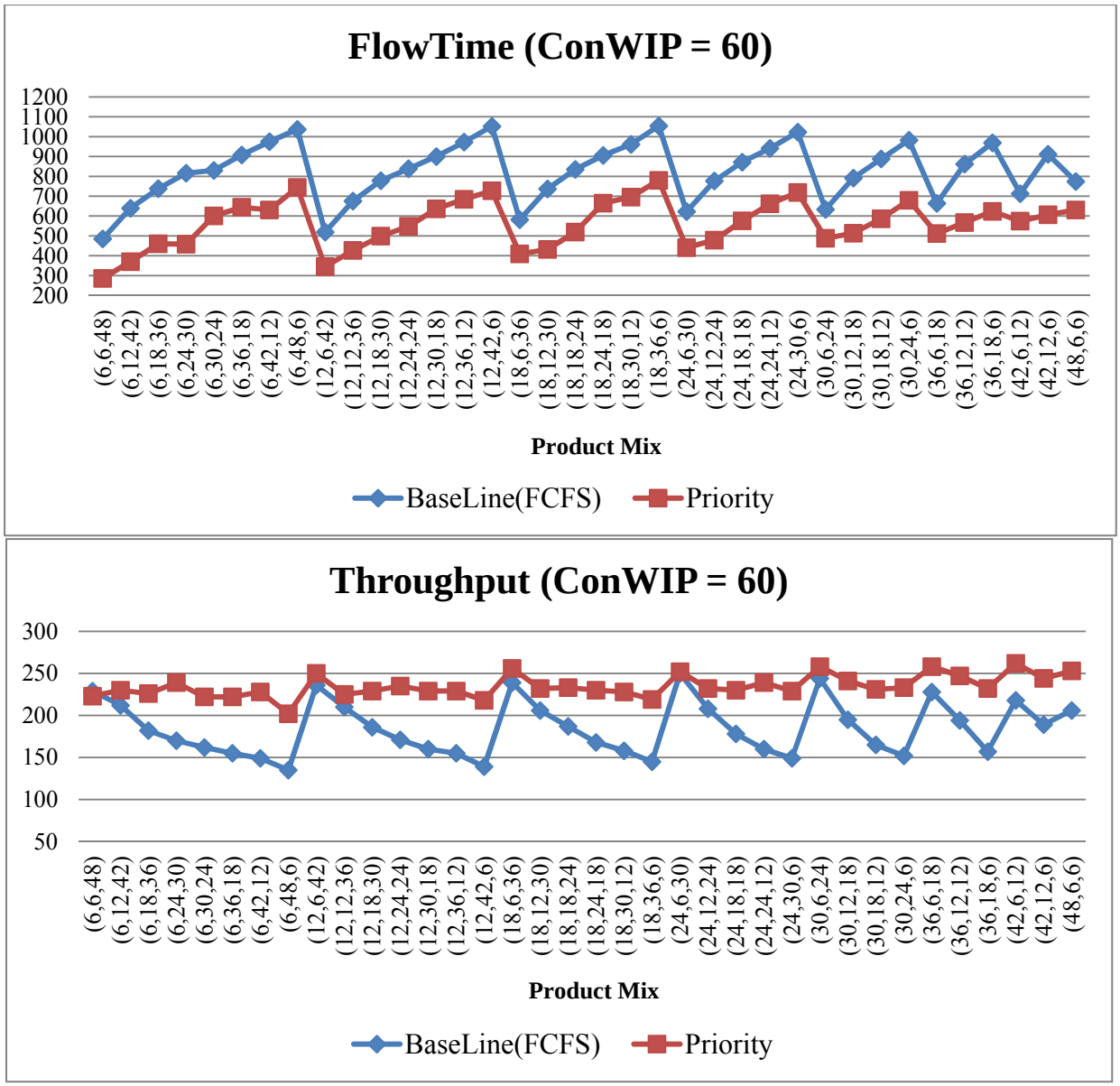


Figure 3.8b Numerical Results 1: Baseline (FCFS) vs. Priority (ConWIP = 60)

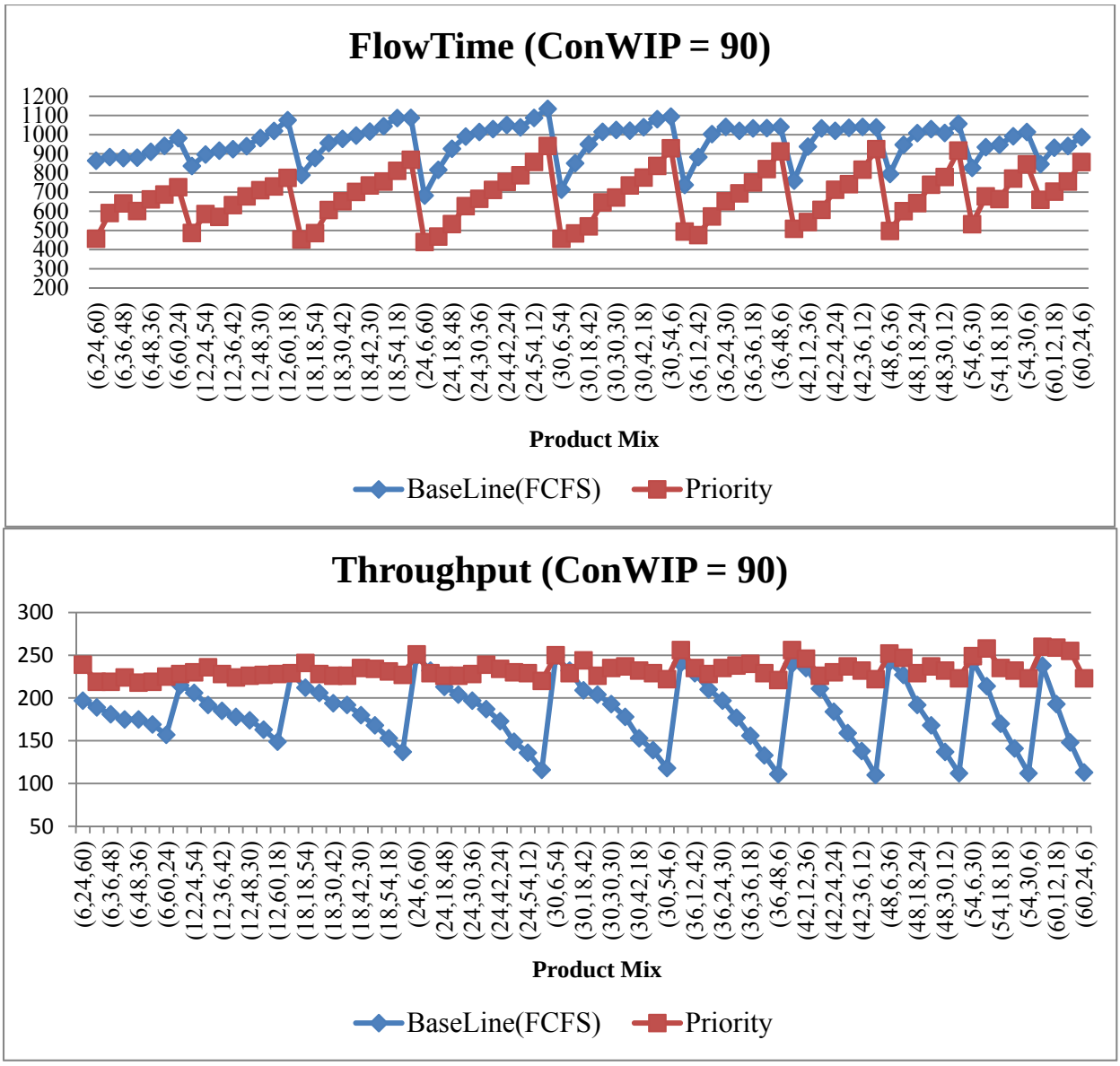


Figure 3.8c Numerical Results 1: Baseline (FCFS) vs. Priority (ConWIP = 90)

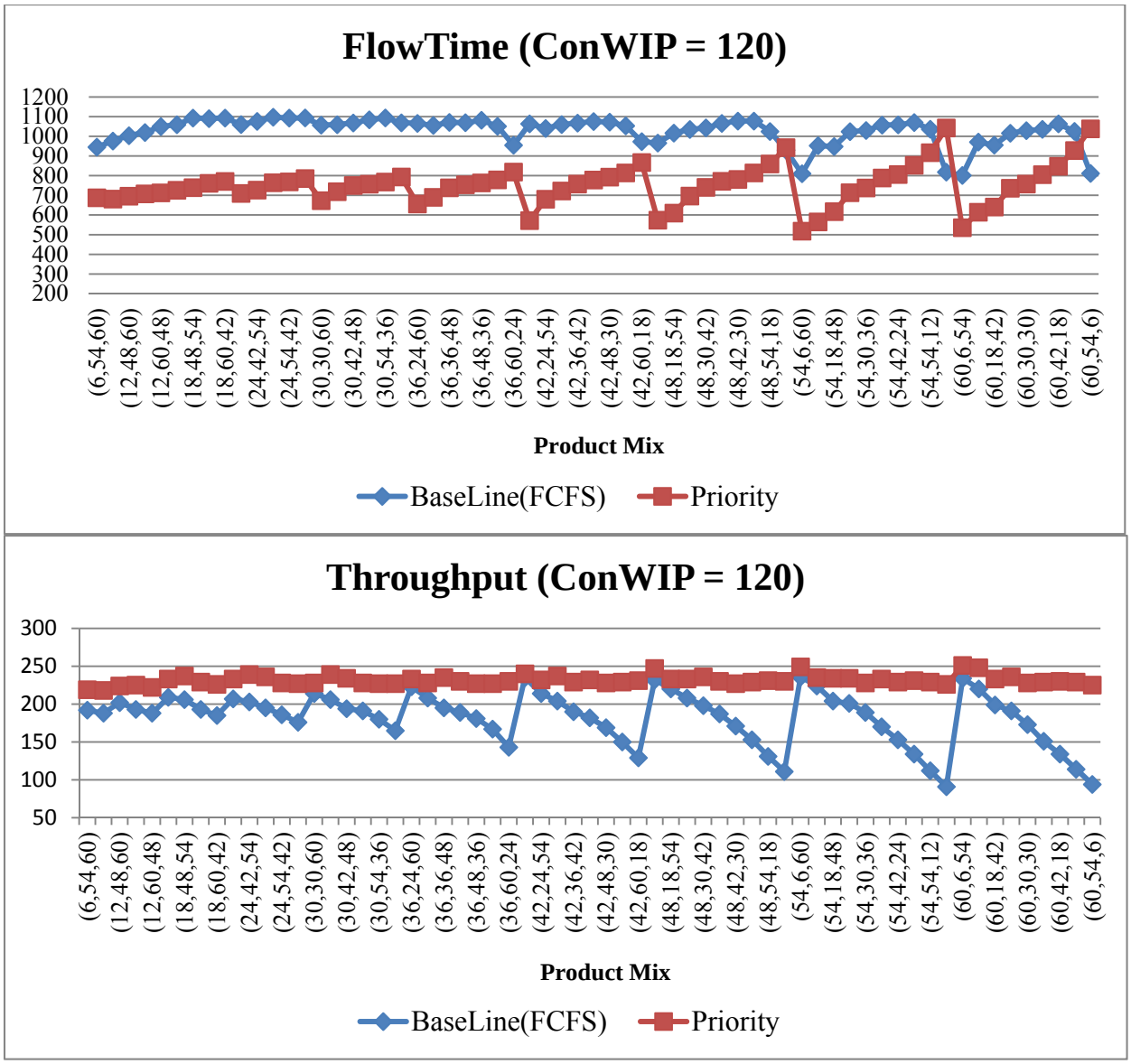


Figure 3.8d Numerical Results 1: Baseline (FCFS) vs. Priority (ConWIP = 120)

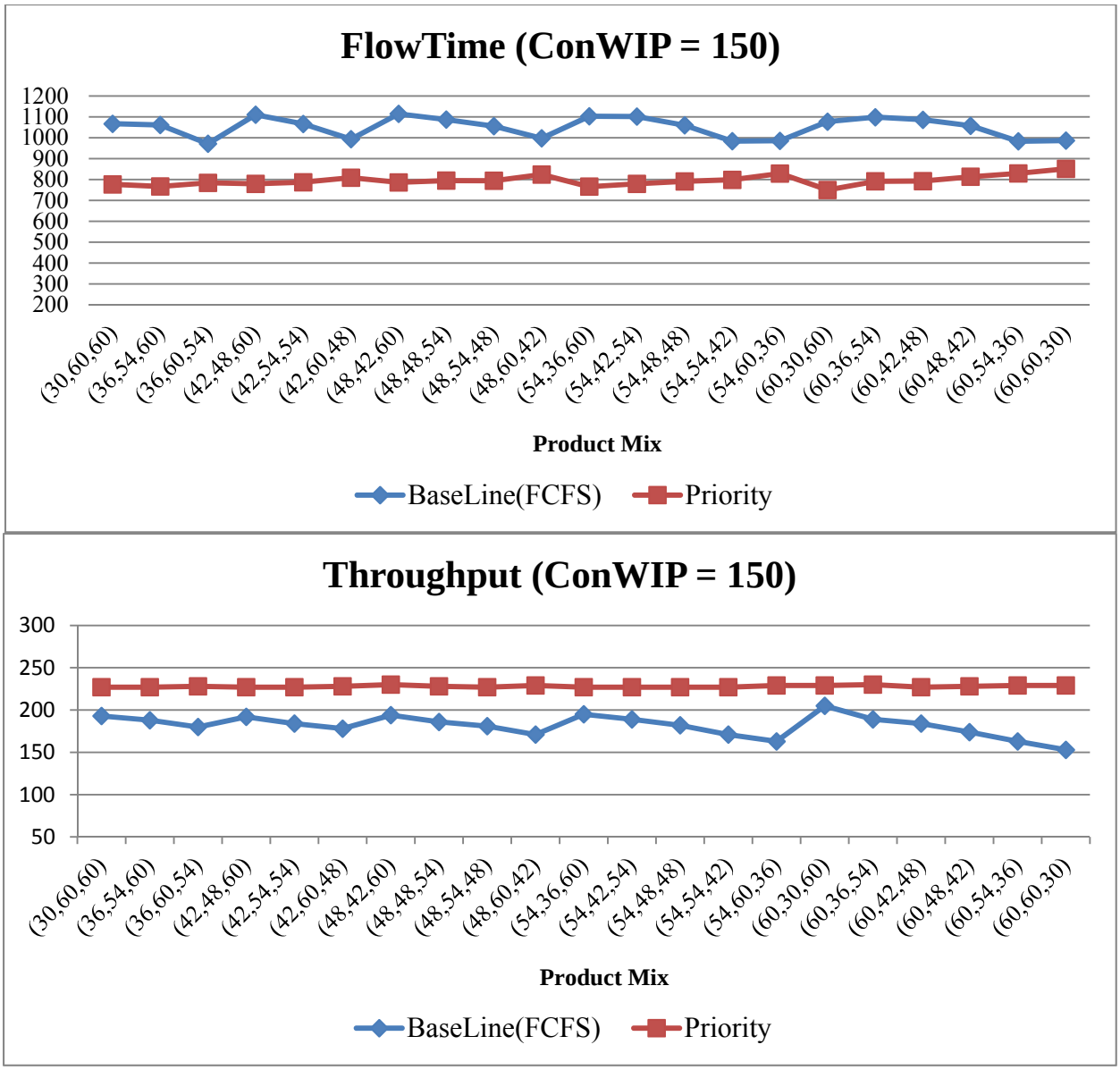


Figure 3.8e Numerical Results 1: Baseline (FCFS) vs. Priority (ConWIP = 150)

3.6.2 Numerical Example 2

To further test the model, in example 2, we assume that for each engine type, there is only one part fails with the probability of 100% and each defective part will pass through all six machines. Of course, the order of processing may vary. For example, for the failed part of Engine Type 1, the sequence may be $M_1 \rightarrow M_2 \rightarrow M_3 \rightarrow M_6 \rightarrow M_4 \rightarrow M_5$, whereas for the failed part of Engine Type 2, the sequence can be $M_1 \rightarrow M_3 \rightarrow M_5 \rightarrow M_4 \rightarrow M_2 \rightarrow M_6$. The configuration of this example is:

Table 3.2 Configuration of Numerical Example 2

Machine	V (Visit Ratio)			S (Service Time)			WL (Workload) = V*S			
	Eng. 1	Eng. 2	Eng. 3	Eng. 1	Eng. 2	Eng. 3	Eng. 1	Eng. 2	Eng. 3	Sum
Machine 1	1	1	1	3	4	5	3	4	5	12
Machine 2	1	1	1	5	7	4	5	7	4	16
Machine 3	1	1	1	3	5	2	3	5	2	10
Machine 4	1	1	1	5	6	8	5	6	8	19
Machine 5	1	1	1	1	4	2	1	4	2	7
Machine 6	1	1	1	3	6	5	3	6	5	14

Based on our definition, machine 4 is the bottleneck machine, *i.e.*, **B** = 4.

The experiment results are illustrated below:

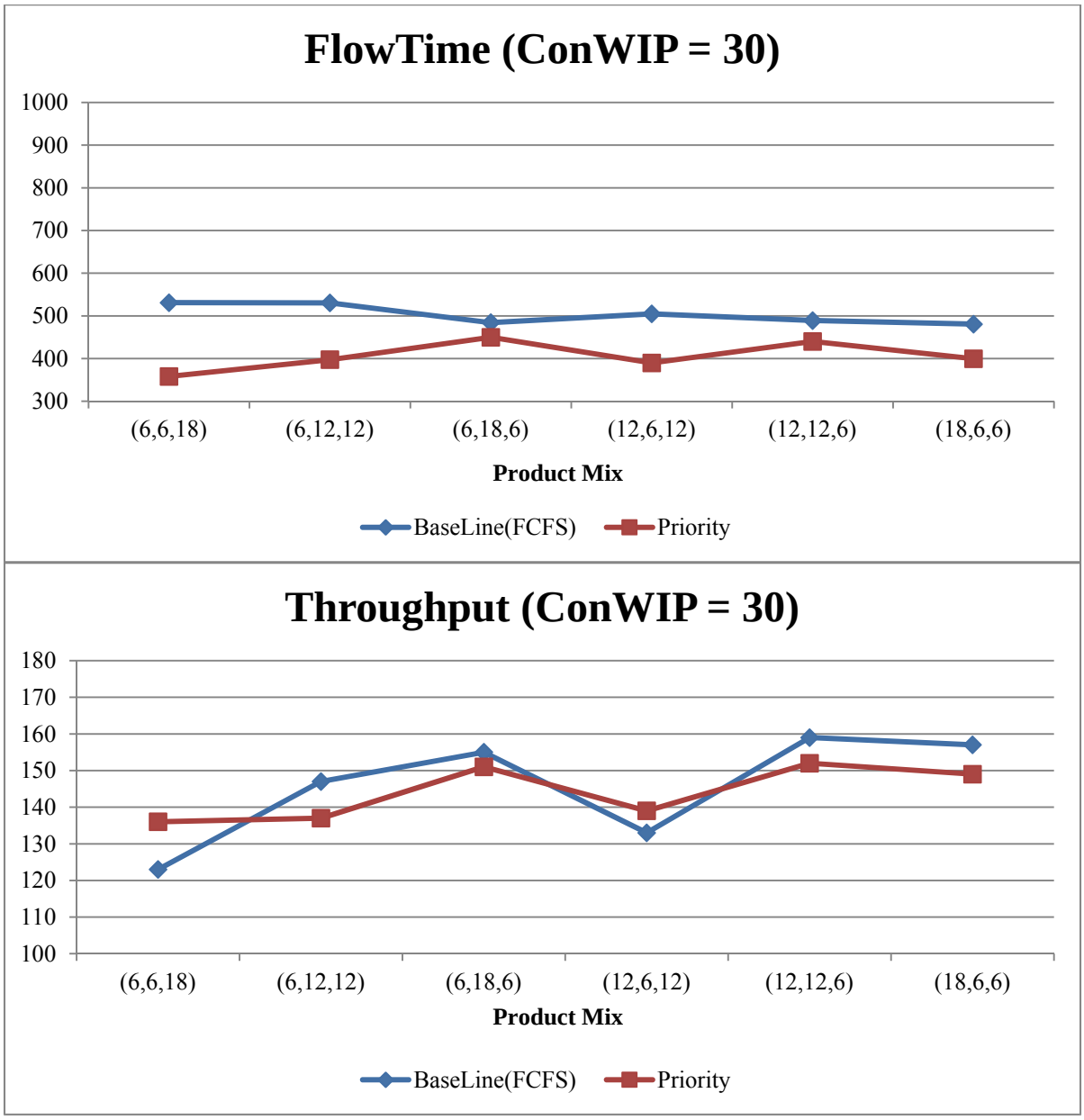


Figure 3.9a Numerical Results 2: Baseline (FCFS) vs. Priority (ConWIP = 30)

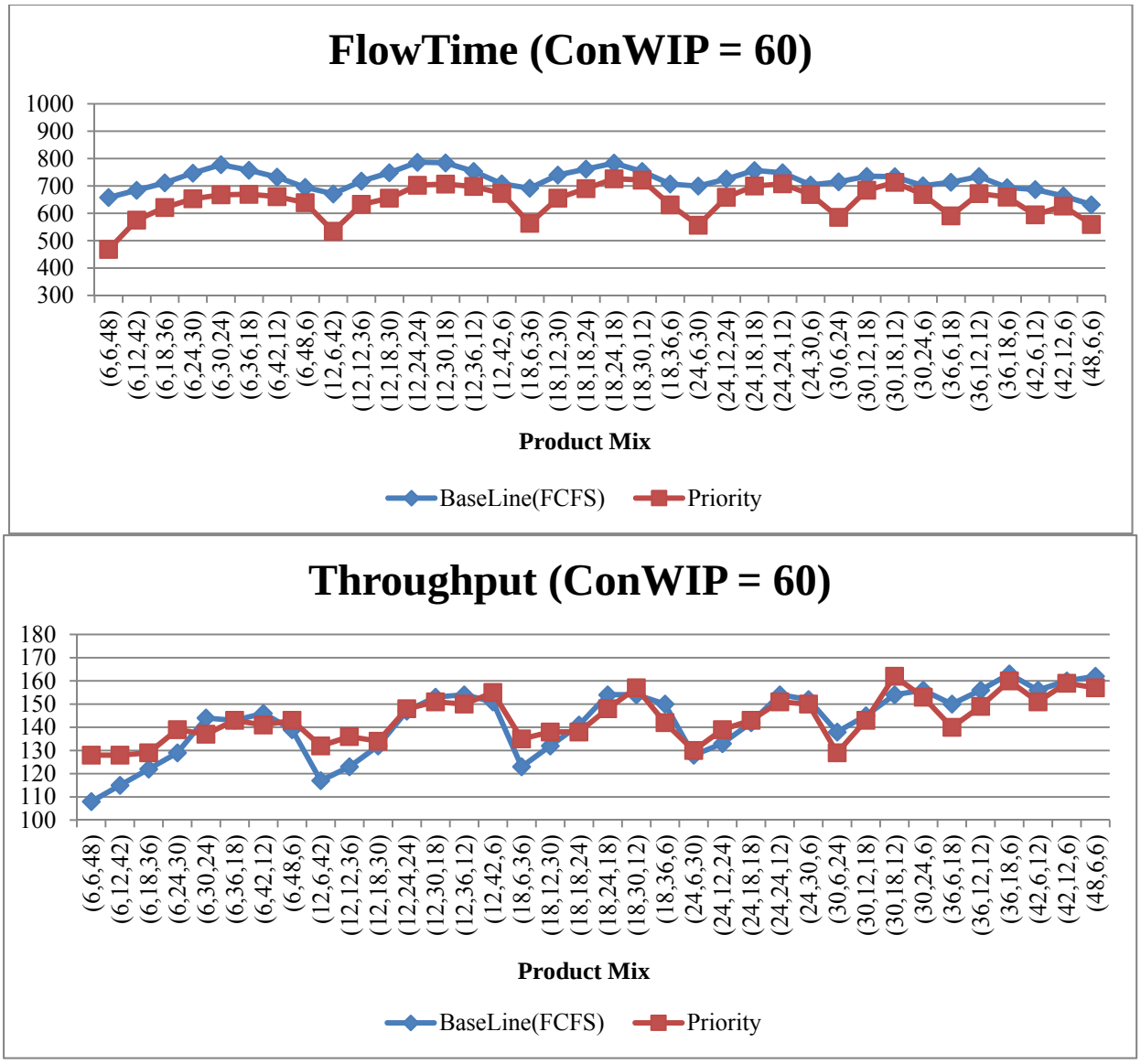


Figure 3.9b Numerical Results 2: Baseline (FCFS) vs. Priority (ConWIP = 60)

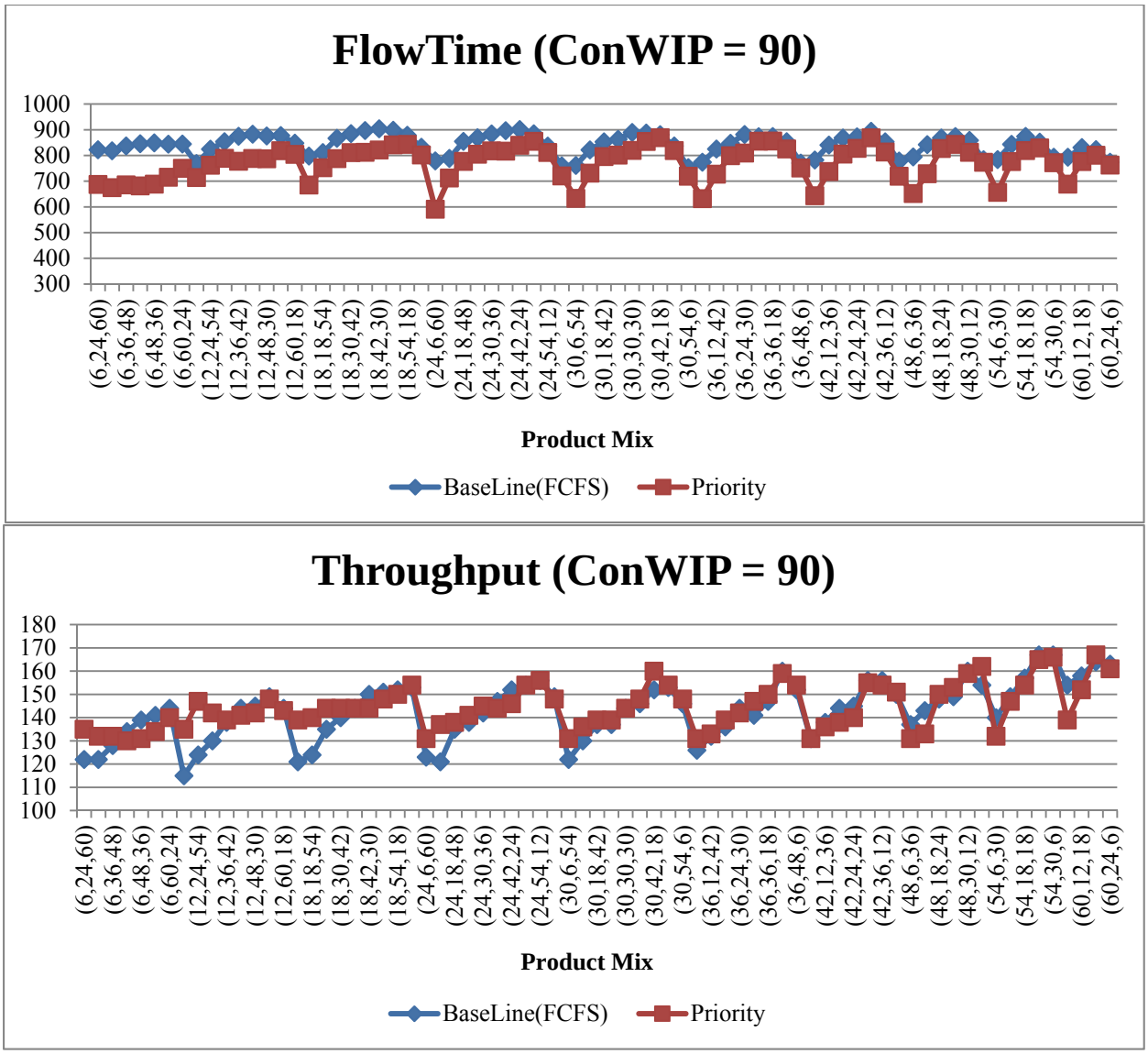


Figure 3.9c Numerical Results 2: Baseline (FCFS) vs. Priority (ConWIP = 90)

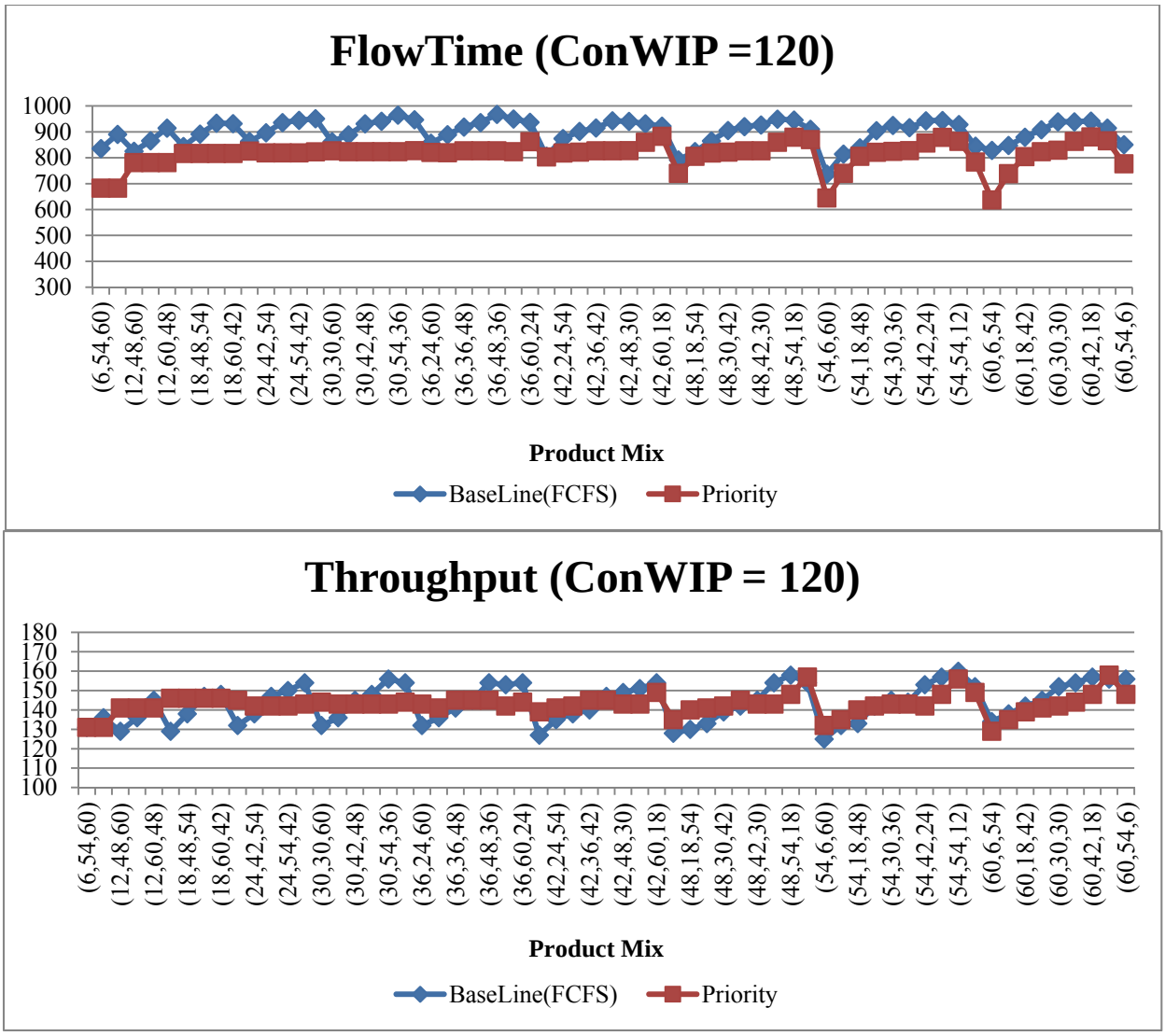


Figure 3.9d Numerical Results 2: Baseline (FCFS) vs. Priority (ConWIP = 120)

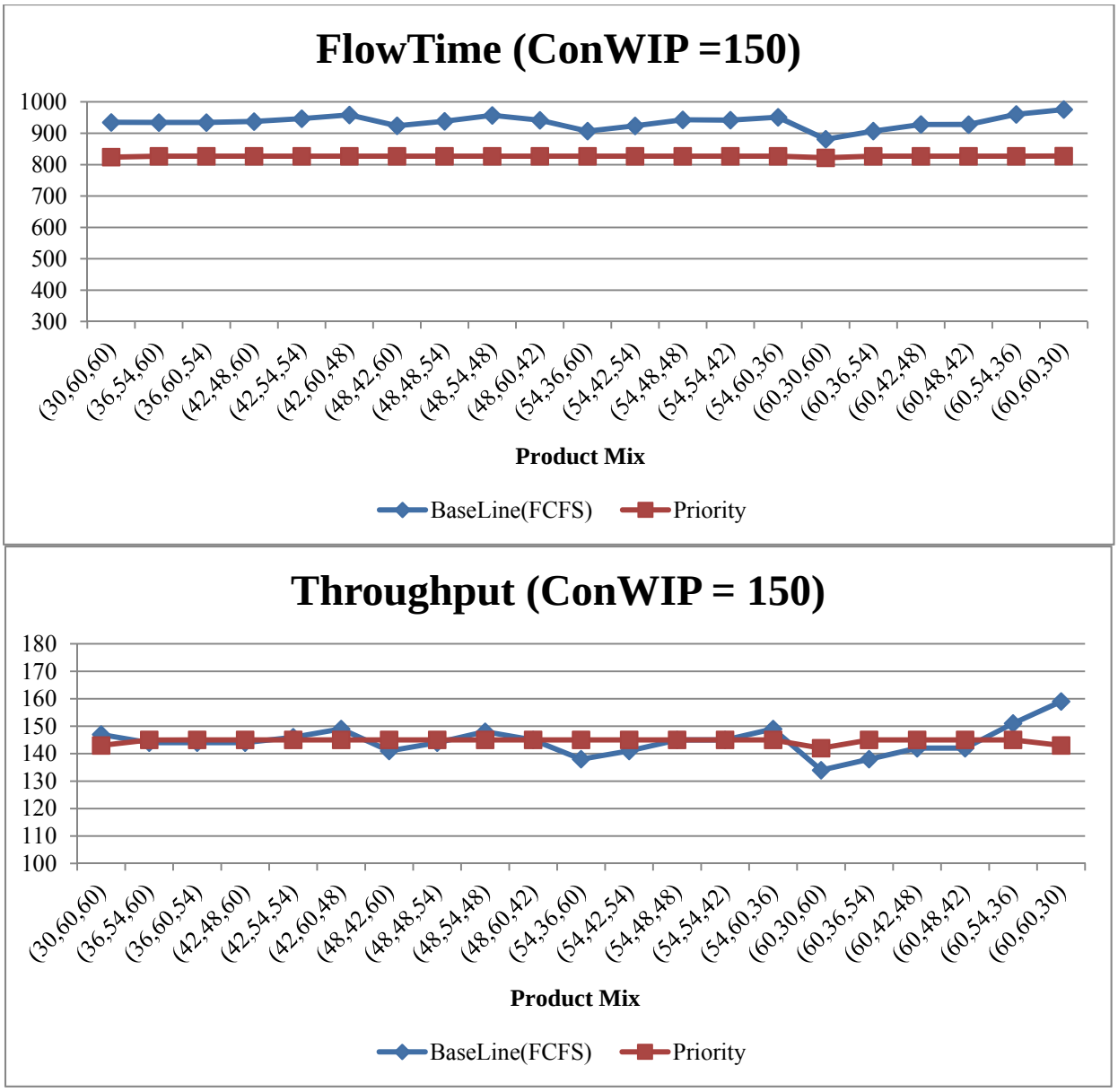


Figure 3.9e Numerical Results 2: Baseline (FCFS) vs. Priority (ConWIP = 150)

3.6.3 Numerical Example 3

This example is much more complicated than the previous ones. All parts of the three Engine Types fail with probabilities P_{ji}, where j =1, 2, 3 and i = A, B, C. Note that, in this example, weighted overall workload ($WL = V*S*P_{ji}$) is used to identify the bottleneck machine. The configuration is presented in Table 3.3.

Based on our definition, machine 1 is the bottleneck machine, *i.e.*, **B** = 1.

Table 3.3 Configuration of Numerical Example 3

Machine	V (Visit Ratio)									S (Service time)									Fairlure Rate								
	Engine Type 1			Engine Type 2			Engine Type 3			Engine Type 1			Engine Type 2			Engine Type 3			Engine Type 1			Engine Type 2			Engine Type 3		
	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C
Machine 1	1	1			2	1	1			4	5			6	3	3			60%	70%	20%	10%	50%	90%	40%	30%	60%
Machine 2		1		1	1		1	1			2		6	4		4	3										
Machine 3	1		1			1	1	1	1	3		3			5	6	4	2									
Machine 4	1	1		1	1	1	1	1		5	3		4	2	4	1	6										
Machine 5			1	1					1			4	7					5									
Machine 6	1		1	1		1			1	4		5	3		6			4									

Machine	WL (Workload) = V*S										Weighted WL (Workload) = V*S*Failure Rate									
	Engine Type 1			Engine Type 2			Engine Type 3			Sum	Engine Type 1			Engine Type 2			Engine Type 3			Sum
	Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C		Part A	Part B	Part C	Part A	Part B	Part C	Part A	Part B	Part C	
Machine 1	4	5	0	0	12	3	3	0	0	27	2.4	3.5	0	0	6	2.7	1.2	0	0	15.8
Machine 2	0	2	0	6	4	0	4	3	0	19	0	1.4	0	0.6	2	0	1.6	0.9	0	6.5
Machine 3	3	0	3	0	0	5	6	4	2	23	1.8	0	0.6	0	0	4.5	2.4	1.2	1.2	11.7
Machine 4	5	3	0	4	2	4	1	6	0	25	3	2.1	0	0.4	1	3.6	0.4	1.8	0	12.3
Machine 5	0	0	4	7	0	0	0	0	5	16	0	0	0.8	0.7	0	0	0	0	3	4.5
Machine 6	4	0	5	3	0	6	0	0	4	22	2.4	0	1	0.3	0	5.4	0	0	2.4	11.5

The experiment results are illustrated below:

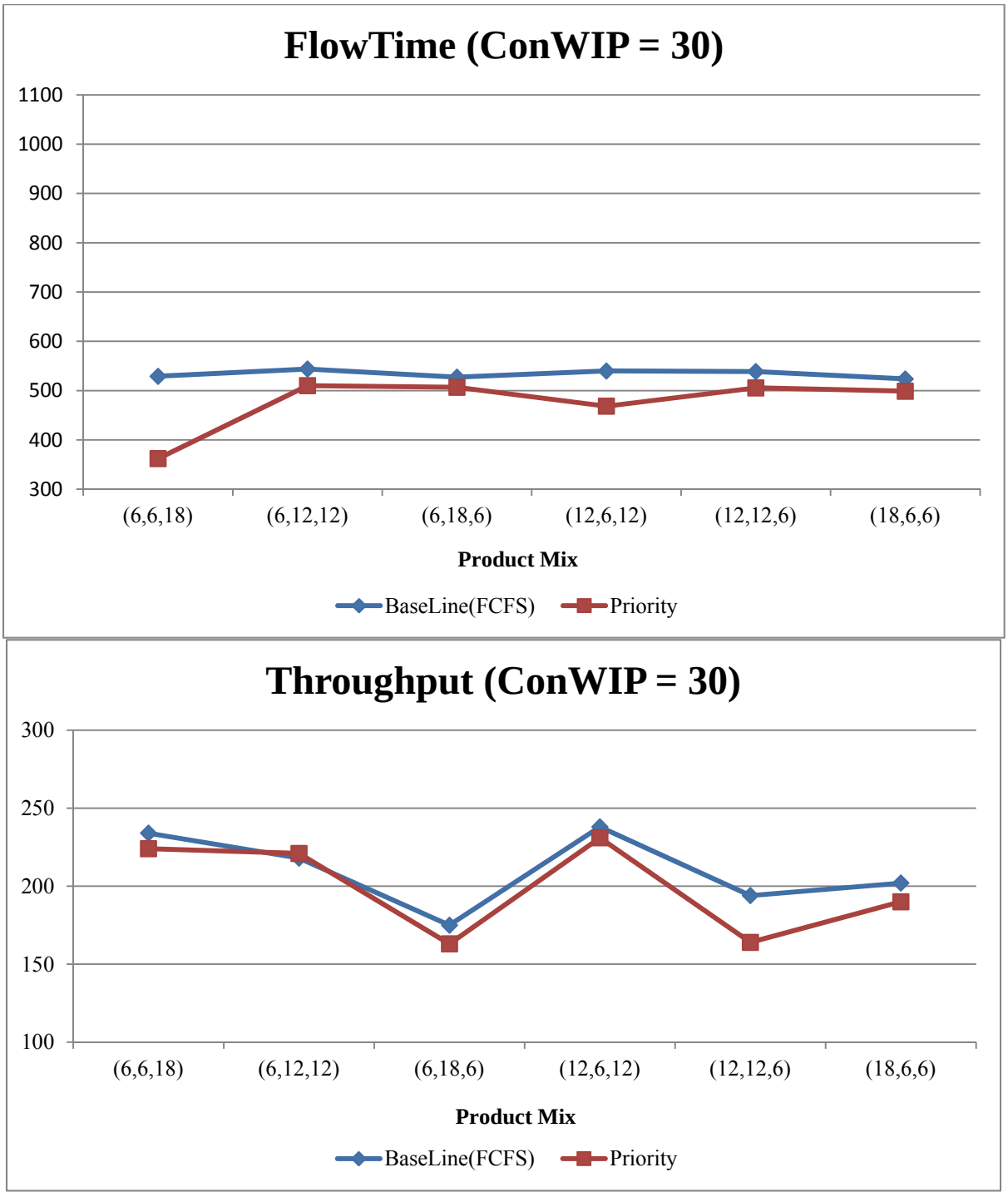


Figure 3.10a Numerical Results 3: Baseline (FCFS) vs. Priority (ConWIP = 30)

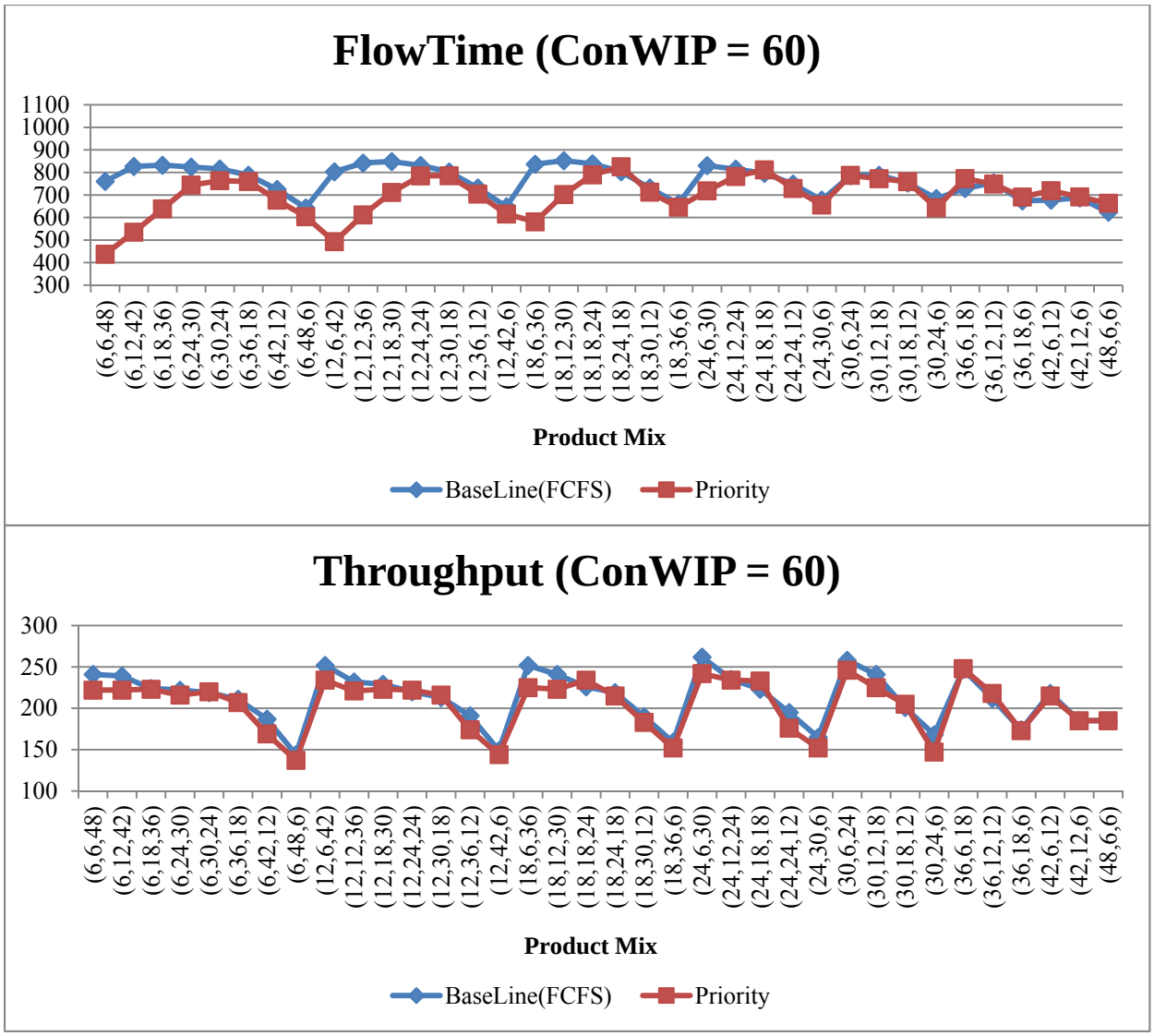


Figure 3.10b Numerical Results 3: Baseline (FCFS) vs. Priority (ConWIP = 60)

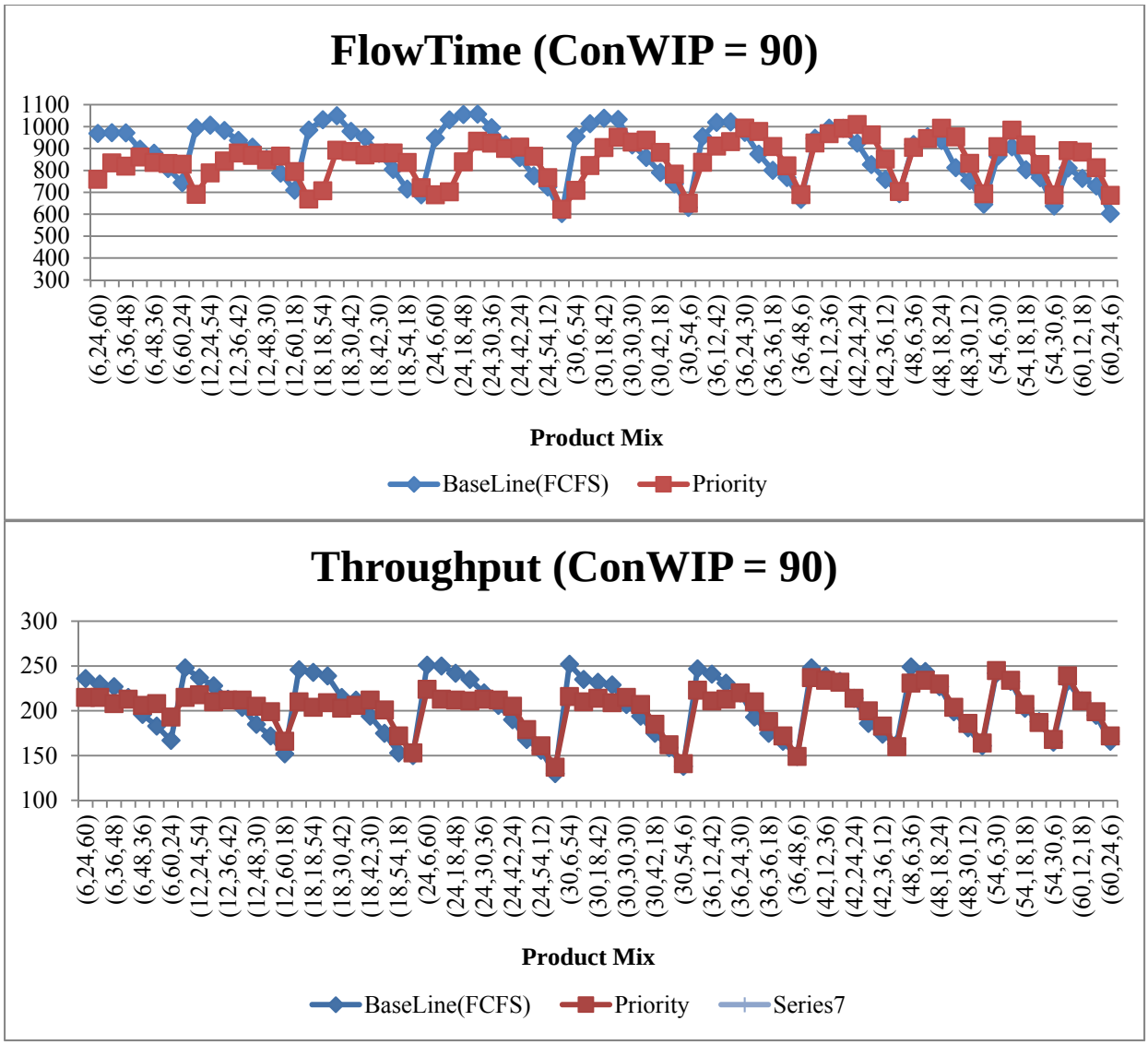


Figure 3.10c Numerical Results 3: Baseline (FCFS) vs. Priority (ConWIP = 90)

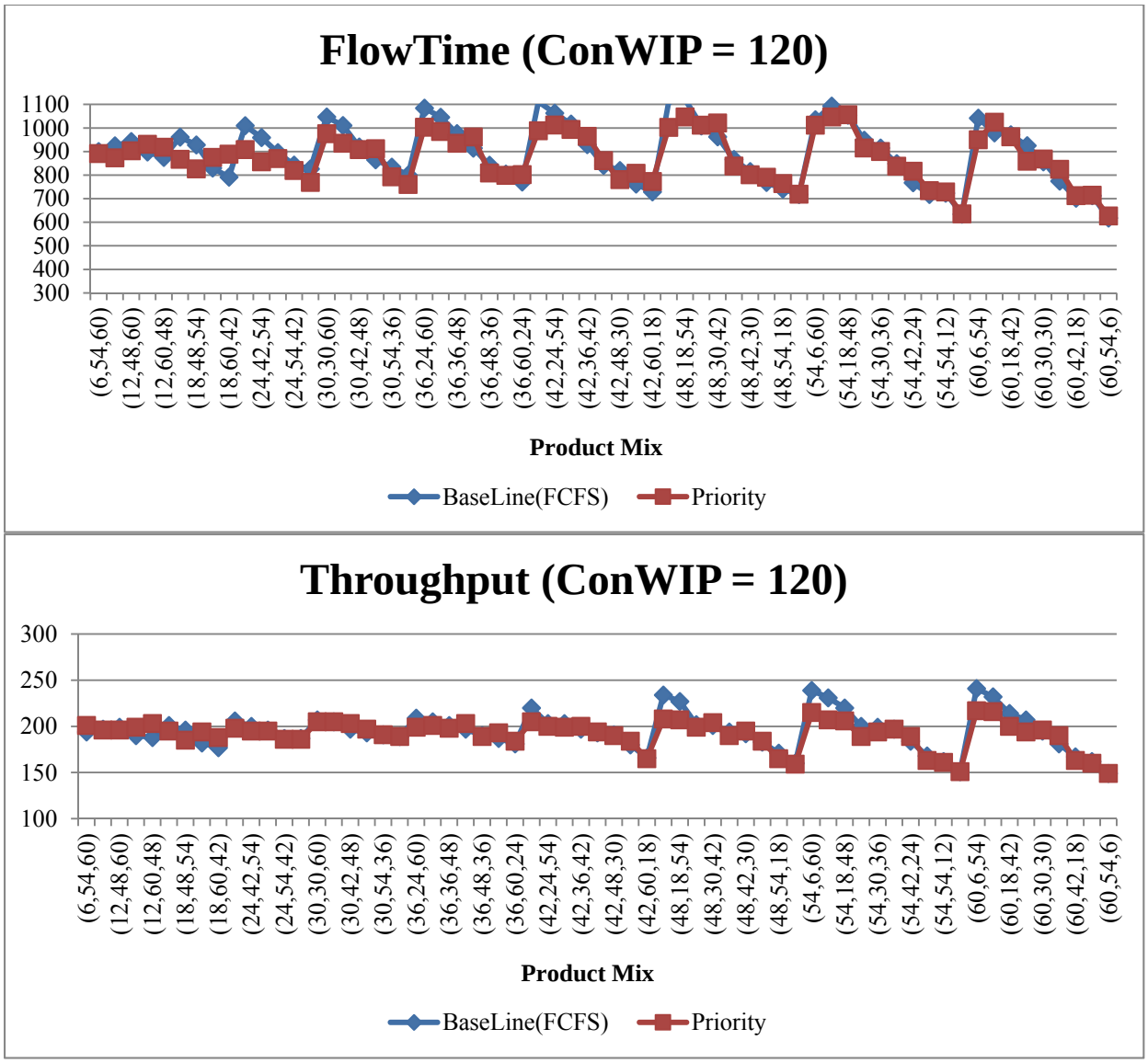


Figure 3.10d Numerical Results 3: Baseline (FCFS) vs. Priority (ConWIP = 120)

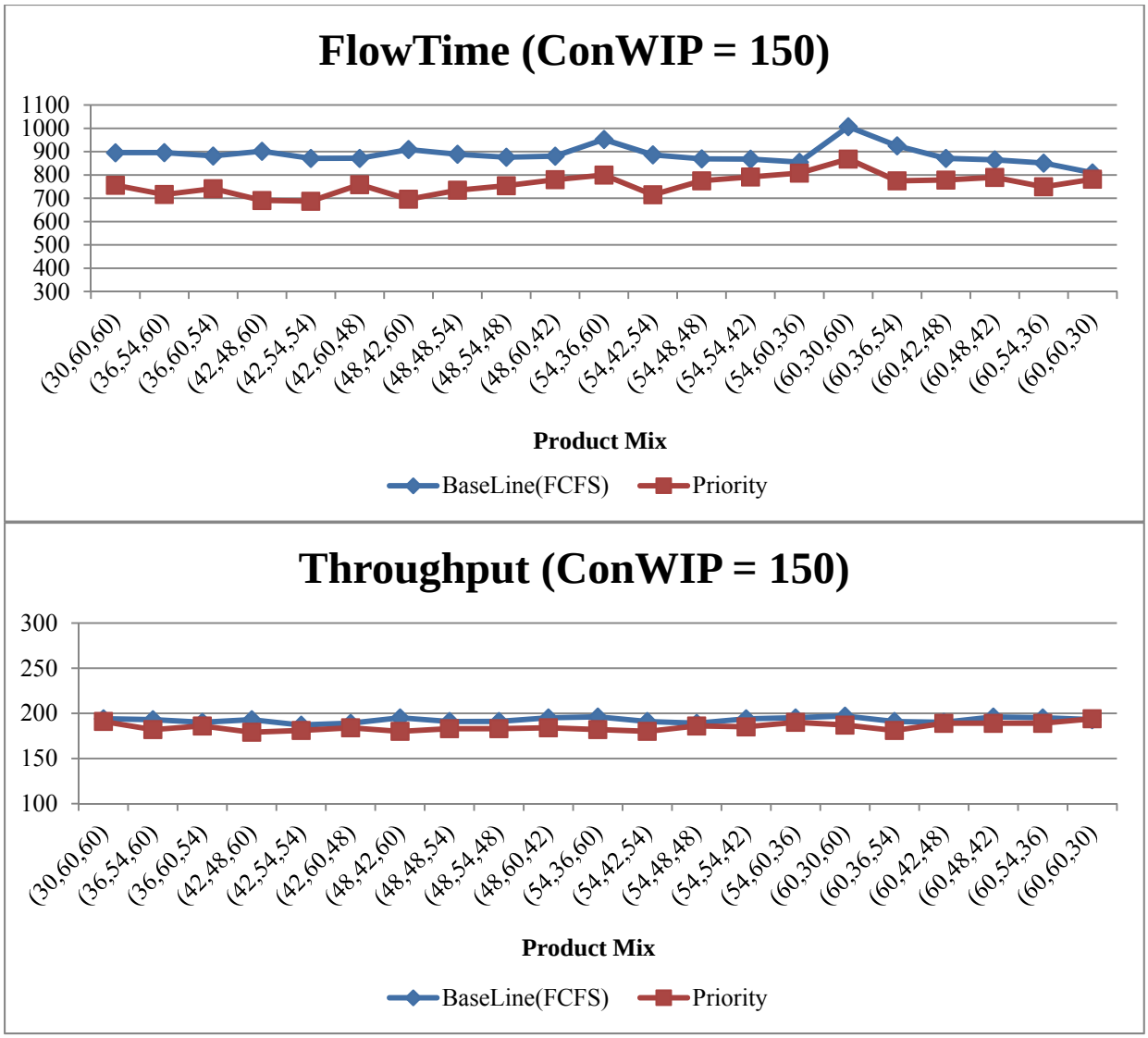


Figure 3.10e Numerical Results 3: Baseline (FCFS) vs. Priority (ConWIP = 150)

3.7 Summary

In this chapter, we conducted a number of experiments using the proposed methodology and demonstrated that the methodology indeed meets the research objective. That is, disassembled parts with the lowest cycle time receive the highest priority on the bottleneck machine and lowest priority on non-bottleneck machines. Simulation results show that this priority scheme increases the system performance by lowering the total average flow time without adversely impacting the total average throughput.

Furthermore, from the simulation experiment, we observed that the model is quite robust. First, the desired results are achieved when the total WIP varies from 30 to 150. Second, as shown in example 3, the priority scheme still works even using the weighted average workload.

CHAPTER 4

PRIORITY QUEUEING NETWORKS

This chapter is organized as follows: in section 4.1, we briefly introduce priority queues, including the associated conservation laws and various service disciplines. Section 4.2 is a literature review of analytical models for mixed priority queueing networks and their solution methods. Section 4.3 provides preliminaries for the discussion of the proposed analytical models. Then, in Section 4.4 and 4.5 we present the general framework of a two-class, two-serve queueing model, which will be used in the subsequent analysis. Also, in these two sections, we discusses the exact solution methods for FCFS, preemptive and non-preemptive priority disciplines. In the last part, we explore conditions under which non-preemptive solutions can be approximated by preemptive solutions. Under those conditions, our results can be effectively expanded to multiple class priority queueing networks.

4.1 Priority Queues

4.1.1 Conservation Laws

The conservation law [Kleinrock (1965), Schrage (1970), Heyman and Sobel (1984)] states that the workload of the priority queueing systems at every instant of time remains unchanged over all work-conserving service-scheduling disciplines. Therefore, as long as there is work in the queue, the server is never idle [Reiser and Kobayashi (1975)], and no job leaves the system before its service is completed. FCFS is a work-conserving queueing discipline.

In multi-class priority queueing systems, the mean waiting time of customers is dependent on their corresponding priority class. Certain multi-class priority queueing systems are work-conserving. However, the following restrictions must be satisfied:

1. No service facility is idle as long as there are jobs in the queue, *i.e*., scheduling is work-conserving [Reiser and Kobayashi (1975)].

2. No job leaves the system before its service is completed.

3. The distributions of the inter-arrival times and the service times are arbitrary with the restriction that the first moments of both the distributions and the second moment of the service time distribution exist.

4. The service times of the jobs are independent of the queueing discipline.

5. Preemption is allowed only when all jobs have the same exponential service time distribution and preemption is of the type preemptive resume (*the preemption causes no loss or creation of service so that the service for the preempted customer is taken up where it left off*).

6. For GI/G/m queues all classes have the same service times. This restriction is not necessary for GI/G/l queues [Heyman and Sobel (1984)]. If for GI/G/m queues the service times of the classes differ, the conservation law is an approximation.

4.1.2 Multiple-class Queueing Models with Priority

I. Introduction

For a multi-class, multi-server queueing system with priorities, several attributes must be clarified. First, the order that the customers are served from the queue must be specified. Usually, the highest priority will be granted to the customer who has a relatively higher measure of importance. Second, the manner that the priority scheme is specified. Here, we restrict our decision of selecting the next available customer for service to be *exogenous*. That is, it depends only on the priority class to which it belongs.

When analyzing multi-class, multi-server closed priority queueing networks, an important rule to follow is *work-conserving law*. According to Gautam (2012), all arriving customers will

eventually complete service and exit from the system and there are no lost customers, *i.e.*, the number of arrivals equals the number of departures. All the service disciplines that we discuss in this thesis are work-conserving.

II. Service Disciplines Based on Priority Types

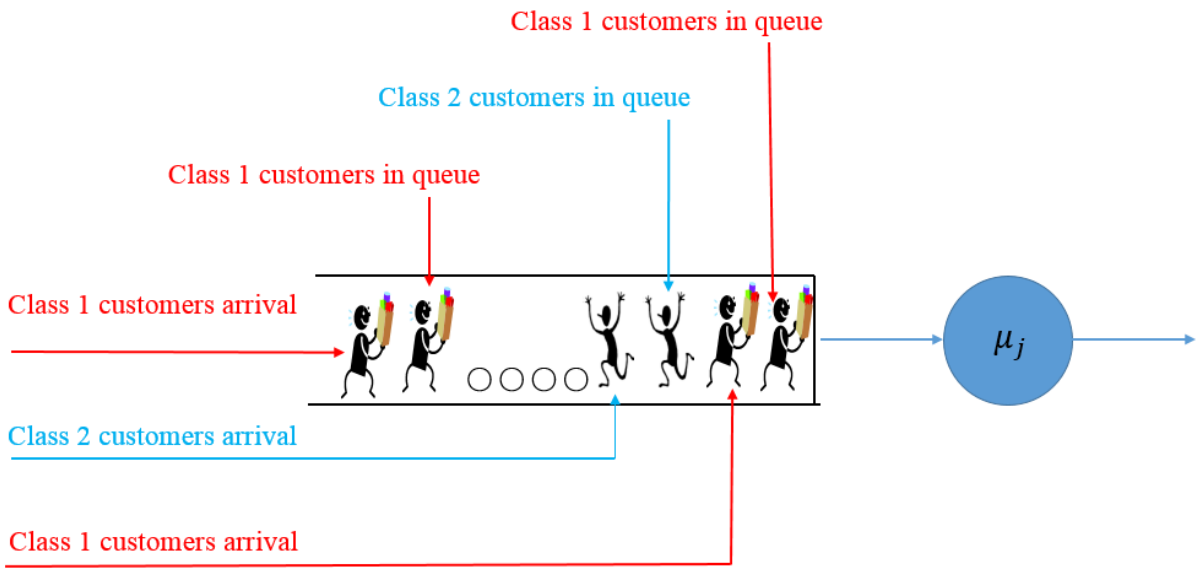
1. First Come, First Serve (FCFS)

The FCFS service-scheduling scheme is quite self-explanatory as the customers are severed in the order of their arrivals regardless of their classes. In this paper, the model using FCFS is also called the *baseline* model, since it is used as a benchmark for other priority schemes. Although this is the simplest scheduling discipline for closed queueing models that have multiple classes, it is difficult to model them analytically. A closed-form solution exists only under certain conditions. For example, if all the servers in the network have class-independent capacities, the closed queueing model with multiple classes will have a product-form solution. Algorithms for solving such systems have been extensively studied [Kant (1992)]. The exact solution technique we use here is Mean Value Analysis (MVA). The MVA method, developed by Reiser and Lavenberg (1980), is based on two simple laws: *Little's theorem* (the mean number of jobs is the product of the throughput and the mean response time of a server or the whole system) and *the arrival theorem* (in a closed product-form queueing network, the *pmf* of the number of jobs seen at the time of arrival to a server i when there are k jobs in the network is equal to the *pmf* of the number of jobs at this server with one less job in the network, which is $k-1$). The method allows us to compute the mean values of measures of interest such as the mean waiting time, throughput, and the mean number of jobs at each server. For detailed introduction of the MVA

algorithm for multi-class, closed queueing networks, see Lazowska (1984), Kant(1992), and Bolch (2006).

2. Non-preemptive (Head-of-the-Line) Priorities

In this service-scheduling scheme, the customers are classified and served according to a priority rule. We assume that the server knows the order of arrivals and class of each customer in the system. Under the non-preemptive priority discipline (also known as Head-of-the-Line (HOL)), once servicing of any ordinary customer begins, it cannot be interrupted by the arrival of any arriving customer. Thus, essentially it is only in the waiting room where the priority has an impact while within a given class, customers are served in FCFS order. Upon arrival, a customer joins the queue ahead of all customers whose priority is lower than that of the arriving customer and behind all customers whose priority is at least as high as the arriving customer. Upon service completion of a unit, the server begins to serve the priority customer who is now at the head of the line (Figure 4.1).



Note: The customer in Class 1 has a higher priority than the customer in Class 2.

Figure 4.1 A Two-Class Priority Queue with HOL Service Discipline

Clearly, FCFS is a special case of non-preemptive.

We assume that the class that should get the highest priority, second highest, etc. is given. This may be obvious in some settings such as a hospital emergency room. However, in other settings, such as a manufacturing system, we may need to determine an optimal way of assigning priorities to different types of jobs.

3. Preemptive Priorities

Preemptive priorities occur when during the service of a customer, a customer with a higher priority arrives, the service of the current customer (with lower priority) is immediately interrupted ("*preempted*"), and the server begins to serve the new customer (with higher priority). Preemptive priorities can be further broken down into two scenarios: preemptive resume and preemptive repeat. In the preemptive resume case, once there are no higher priority customers in the system, the server resumes the processing of the interrupted customer at the point where it was interrupted. In the preemptive repeat case, once there are no higher priority customers in the

system, the interrupted customer has to start the service from the beginning again. Note that, because all the service times presented in this thesis are exponentially distributed, the results for preemptive resume and preemptive repeat are identical due to the memoryless property. Therefore, there is no difference between these two priority rules.

Similarly, the service discipline inside the priority class is FCFS.

4.2 Related Studies on Mixed Priorities

The above priority service disciplines are standard and "pure" priority schemes, *i.e*., all customers are severed under either preemptive or non-preemptive discipline. However, very often, there are also mixed priorities and class switching in real systems. Typical examples are the models of UNIX-based operating system [Bolch et al. (2006)] and the models of cellular mobile networks [Greiner, Bolch, and Begain (1998)], in which mixed priority strategies are used.

There are two types of mixed priority queueing models in literature: traditional mixed priority models and Morris' models. These models are briefly discussed in the following sections.

4.2.1 Traditional Mixed Priority Models

In most of the literature, mixed priority models refer to systems that combine a preemptive priority discipline with a non-preemptive priority discipline. Often in practice this strategy is used to differentiate classes of customers with different service level agreements. No results for multi-server, multi-class systems exist to our best knowledge.

Jaiswal (1968) and Linus Schrage (1969) analyze a two-class mixed priority model. In the first part of a class 2 processing time, class 1 customers have preemptive-resume priority over the class 2 customer, but after a certain point in the class 2 customer processing time, class 1

customers no longer have preemptive priority over class 2 customers. Further, L. Schrage (1969) partitions a job into an arbitrary number of intervals of two types: non-interruptive portions and interruptible portions. Type 2 intervals can further be classified as being repeat with resampling (drawing randomly with replacement from a set of data points) or repeat without resampling. Setup time is required for a job to regain the processor. No solution is given for determining which intervals should be pre-emptible to optimize the whole system.

Also, a combination of HOL (Head-Of-the-Line) strategy and PR (Processor Sharing) is also found in the literature. This mixed priority scheme is also widely used in UNIX-based operation systems and in models of cellular mobile networks [Greiner et al. (1998)].

4.2.2 Morris' Mixed Priority Models

Morris (1981) develops a preemptive priority queueing model which is used for transmission of messages. Specifically, there are two nodes (servers) and two grades (classes) of messages in the system – the premium grade and the standard grade in the system. Also the order of the priorities are reversed in those two nodes. The general approach Morris uses is to first set up the balance equations (steady-state Kolmogorov forward equations) for the Markov chains describing the number of customers of each priority class at each node. Then he obtains the stationary distribution by solving these equations. Subsequently, mean throughput and mean delay are computed for each customer class. Also, Morris gives an approximate solution to the system with non-preemptive priority at one node.

Rumsewicz and Henderson (1989) extend Morris' work for service time distribution from state independent to state dependent. They also obtain the exact solution for two-node systems with priorities reversed, state dependent service parameters and generalized service time distributions.

In addition, they obtain a matrix geometric solution for two-node systems with preemptive priority discipline at one node and any non-batch servicing queue discipline at the other.

4.3 Preliminaries

Throughout this chapter, we consider a two-class, two-server closed queueing network in which the priority scheme developed in Chapter 3 is applied. In this scheme, we identify the bottleneck machine based on the overall workload and classify machines into two categories: bottleneck machine and non-bottleneck machines. Engine Types (job types) with lower mean cycle time receive higher priority on bottleneck machine and lower priority on non-bottleneck machines. Correspondingly, Engine Types (job types) with higher cycle time receive lower priority on bottleneck machine and higher priority on non-bottleneck machines.

Our goals are to obtain insights into the analytical solution for the simple cases of closed queueing networks with preemptive or non-preemptive priorities, to explore the conditions under which non-preemptive solutions can be approximated by preemptive solutions, and to draw general conclusions regarding the system performance on the effect of some simple network priority structures occurring in practice.

Notation that is frequently used in this chapter is summarized below.

Notation	Description
A	Engine Type (job type) A
B	Engine Type (job type) B
n_A	Number of type A jobs at machine 1
n_B	Number of type B jobs at machine 1
N_A	Number of type A jobs in the ConWIP system
N_B	Number of type B jobs in the ConWIP system
M_1	Machine 1
M_2	Machine 2.
μ_{1A}	Mean service rate for type A jobs at machine 1
μ_{2A}	Mean service rate for type A jobs at machine 2
μ_{1B}	Mean service rate for type B jobs at machine 1
μ_{2B}	Mean service rate for type B jobs at machine 2
C_1	$\mathsf{C}_1 \in \{A, B, 0\}$, type of job in service at machine 1; if $\mathsf{C}_1 = 0$, machine 1 is idle
C_2	$\mathsf{C}_2 \in \{A, B, 0\}$, type of job in service at machine 2; if $\mathsf{C}_2 = 0$, machine 2 is idle

4.4 Two-Class, Two-Server Queueing Model with FCFS

The system considered in this section is shown in Figure 4.2. The system is designed to model a job-shop consisting of two machines with each processing two types of jobs (Engines) using FCFS discipline. There are N_A number of type A jobs and N_B number of type B jobs in the system. At machine i $(i \in \{1,2\})$, the service times are exponentially distributed with parameters μ_{iA} and μ_{iB} for type A jobs and type B jobs respectively. All jobs follow FCFS service discipline at each machine. The system is a ConWIP system. Upon the completion of service at machine 1, the job proceeds immediately to machine 2.

The exact solution technique to derive the system performance measures for this queueing network can been found in Lazowska (1984) and Schwetman (1980). The solution can be obtained through the extension of the single class Mean-Value analysis (MVA) algorithm.

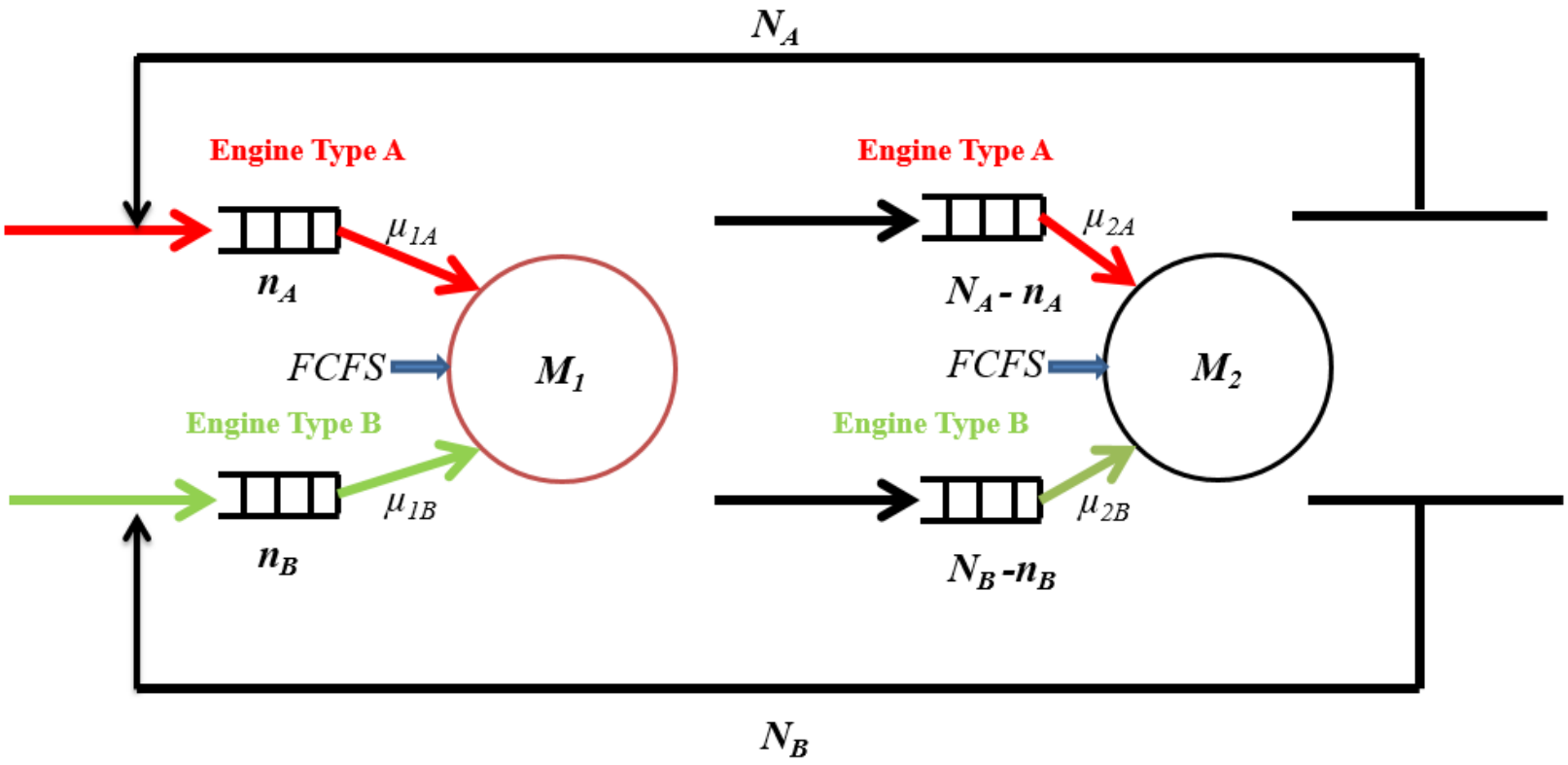


Figure 4.2 Two-Class, Two-Server System with FCFS at Both Servers

4.5 Two-class, Two-Server Queueing Model with Mixed Priorities

4.5.1 General Description

Consider a two-class, two-server closed queueing system shown in Figure 4.3. Figure 4.4 depicts the corresponding *ExtendSim* simulation model. Note that, the priority orders at machine 1 and machine 2 are reversed, *i.e*., type A jobs have a higher priority at machine 1 and have a lower priority at machine 2 and vice versa. There are N_A number of type A jobs and N_B number of type B jobs in the system all the time, *i.e*., N_A and N_B are fixed. The service time distributions at machine i ($i \in \{1,2\}$) are assumed to be exponential with parameters μ_{iA} and μ_{iB} for type A jobs and type B jobs respectively. The service times are not required to be the same at machine i for each type of job. Service within a customer class at machine i follows FCFS discipline. The system can be regarded as having two queues, one for each job type.

The priority scheme used in this chapter is as follows: Engine Type (job type) with lower mean cycle time receive higher priority on the bottleneck machine and lower priority on the non-bottleneck machines. The priority orders are reversed for Engine Type (job type) with higher mean cycle time. For convenience purpose, we assume that machine 1 is the bottleneck machine and job type A has a lower mean cycle time based on FCFS. Therefore, job type A will be granted a higher priority than job type B at machine 1; the priority order will be reversed at machine 2.

The general approach to analyze this model is to set up the balance equations (steady-state Kolmogorov forward equations) for the Markov chain describing the number of job types of each priority class at machine i ($i = 1,2$). The stationary distribution can be obtained by solving these equations and hence the mean throughput and flow time can be computed for each job type.

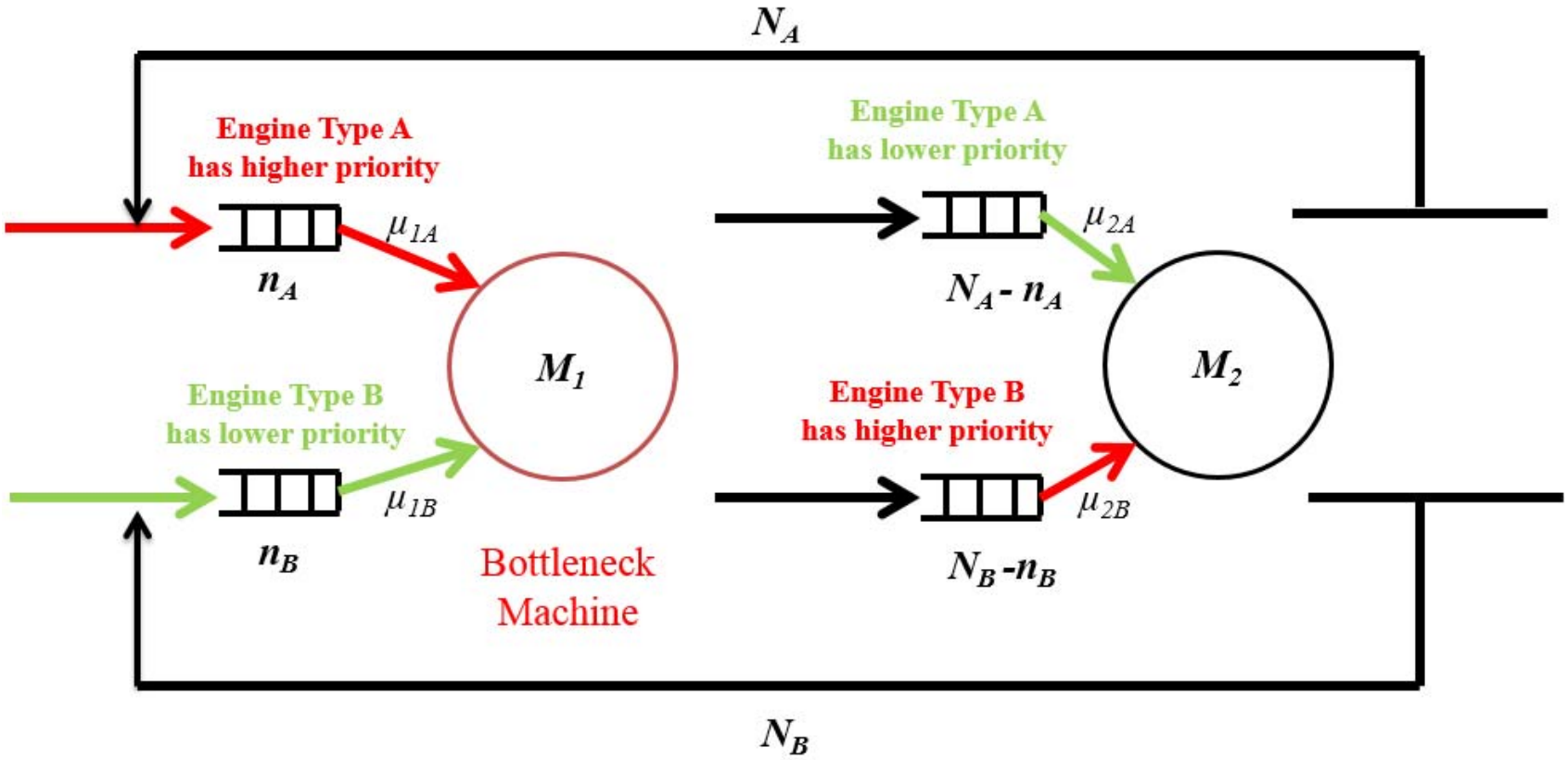


Figure 4.3 Two-Class, Two-Server System with Priorities at both Servers

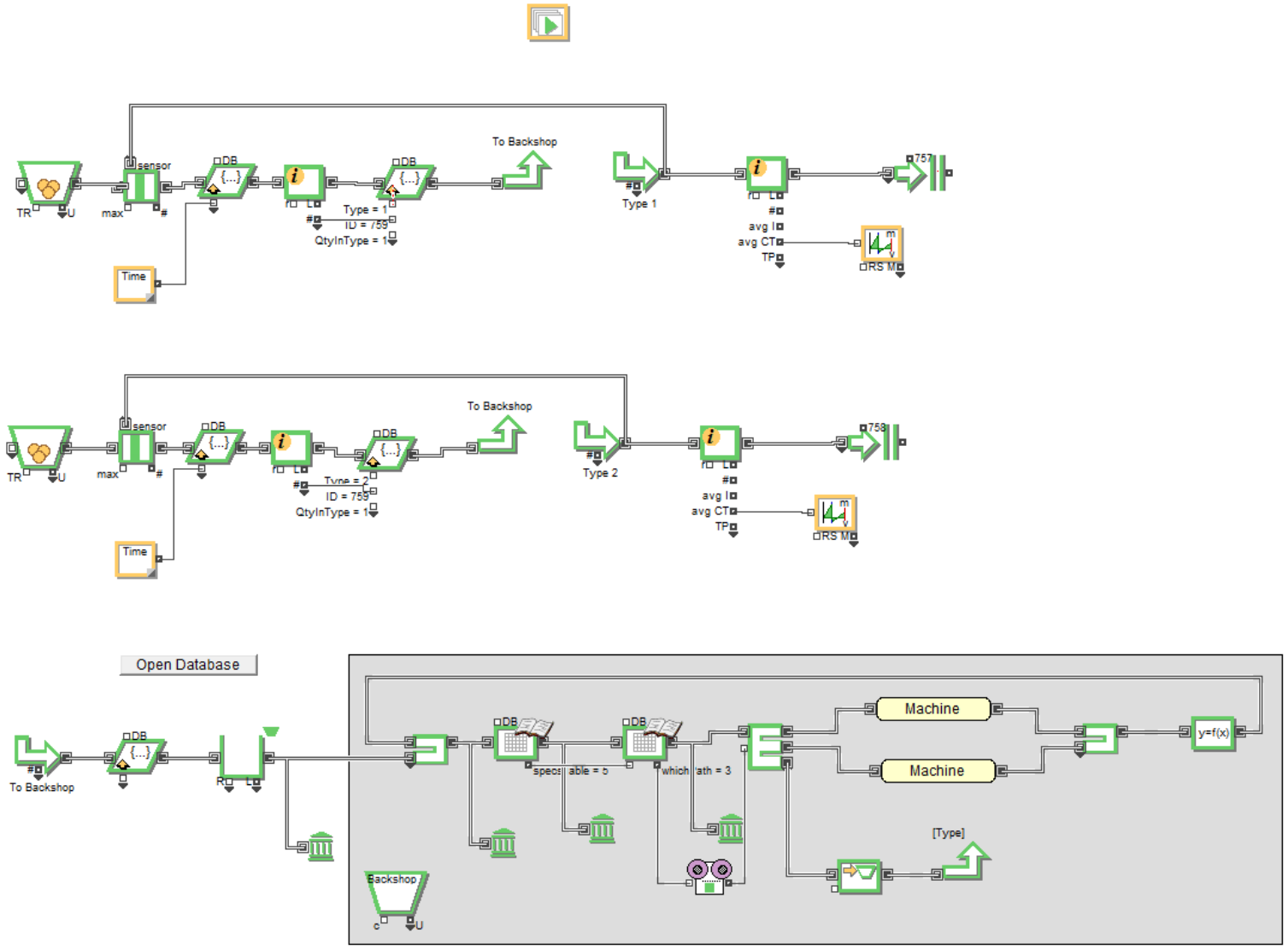


Figure 4.4 Two-Class, Two-Server Simulation System using *ExntendSim*

4.5.2 Preemptive Priorities

Because it is assumed that all service times are exponentially distributed, the priorities can be understood to be either preemptive resume or preemptive restart (with resampling). We describe the state of the system by (n_A, n_B) when there are n_A number of type A jobs and n_B number of type B jobs at machine 1 respectively. Since machine 1 is the bottleneck machine, Engine Type A will received a higher priority at machine 1 and a lower priority at machine 2. For Engine Type B, the priority order is reversed at both machines. Higher priority jobs "preempt" lower priority jobs at each machine. The state transition diagram for $N_A = 2, N_B = 2$ is shown in Figure 4.5 [Morris (1980)].

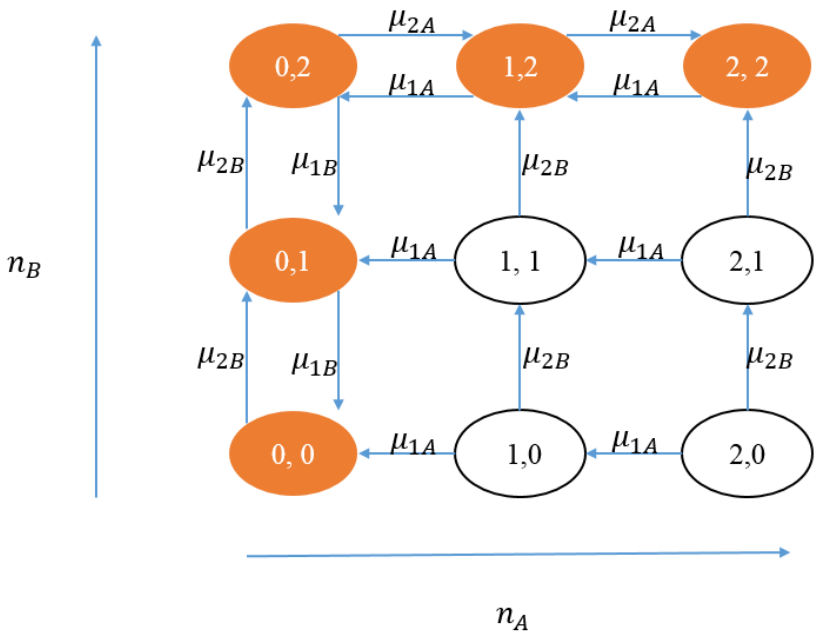


Figure 4.5 State Transition Diagram for Two-Class, Two-Server Mixed Preemptive Priority System $(N_A = 2, N_B = 2)$

It can easily be expanded to (N_A, N_B), where $N_A \geq 2, N_B \geq 2$, as illustrated in Figure 4.6.

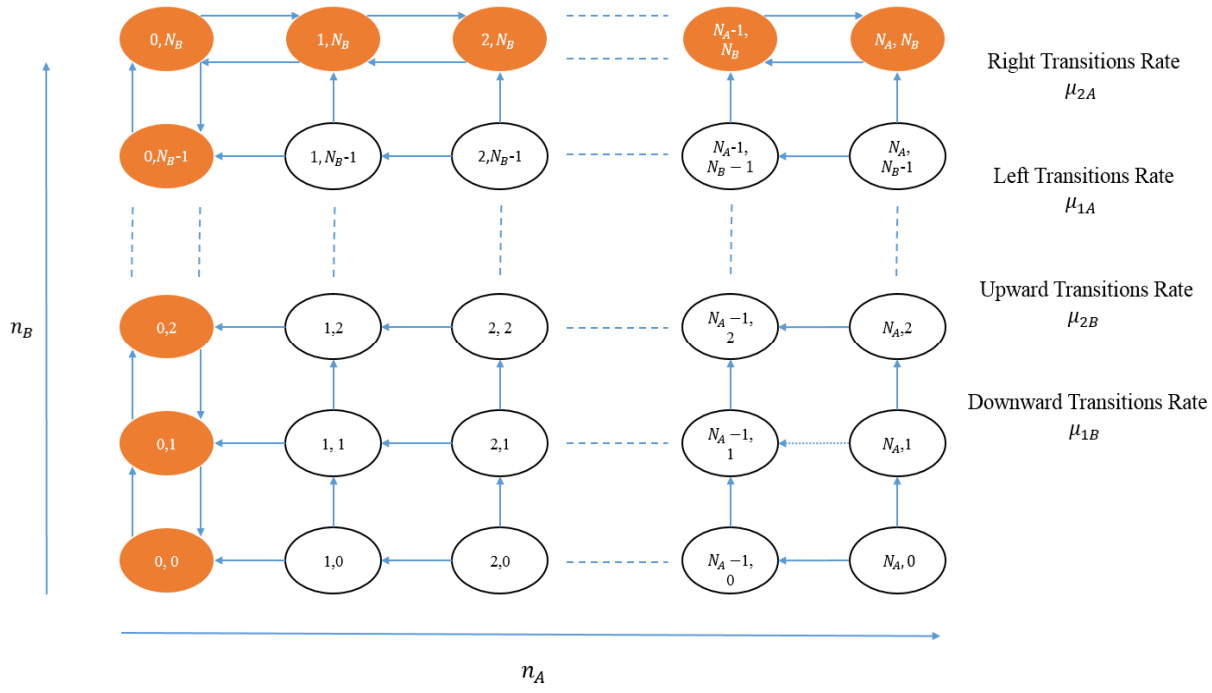


Figure 4.6 State Transition Diagram for Two-Class, Two-Server Mixed Preemptive Priority System

From the above state transition diagrams, we recognize that states $\{(n_A, n_B): n_A > 0, and\ n_B < N_B\}$ are transient, *i.e.*, in equilibrium, one of the two higher priority queues is always empty. This observation has practical implications: first, this mixed priority scheme balances the flows of the system; second, it avoids overloading the bottleneck machine.

States $(0,\ n_B)$ and $(n_A\ , N_B)$ form one recurrent class (see color shaded states in Figure 4.5 and Figure 4.6).

Let $p(n_A, n_B)$ be the stationary probability of state (n_A, n_B). Based on the recurrent states, we have

$$p(0,\ n_B) = P_0 \left(\frac{\mu_{2B}}{\mu_{1B}}\right)^{n_B}, \text{where } n_B = 0,1, \dots,\ N_B$$

$$p(n_A, N_B) = P_0 \left(\frac{\mu_{2B}}{\mu_{1B}}\right)^{N_B} \left(\frac{\mu_{2A}}{\mu_{1A}}\right)^{n_A}, \text{where } n_A = 0,1, \dots, N_A .$$

with

$$P_0 = \frac{1}{\sum_{n_B=0}^{N_B-1}\left(\frac{\mu_{2B}}{\mu_{1B}}\right)^{n_B} + \left(\frac{\mu_{2B}}{\mu_{1B}}\right)^{N_B} \sum_{n_A=0}^{N_A}\left(\frac{\mu_{2A}}{\mu_{1A}}\right)^{n_A}}.$$

The mean overall throughput of type A (respectively type B) jobs denoted by TH_A (respectively TH_B):

$$TH_A = \mu_{1A} \sum_{n_A=1}^{N_A} p(n_A, N_B)$$

$$TH_B = \mu_{1B} \sum_{n_B=1}^{N_B} p(0, n_B).$$

The state probability of the network with multiple job classes is represented by $\pi(\mathbf{S}_1, \dots, \mathbf{S}_N)$. The normalization condition that the sum of the probabilities of all possible states is 1 must be satisfied here.

Let

$$\pi_i(\mathbf{k}) = \text{marginal probability that the } i\text{th machine is in the state } \mathbf{S}_i = \mathbf{k},$$

In this case**,** $\pi_i(\mathbf{k})$ is determined by $p(n_A, N_B)$ and $p(0, n_B)$.

Then the mean number of jobs of type j at machine i is

$$\bar{n}_{ij} = \sum_{\mathbf{k}} n_j . \pi_i(\mathbf{k}), i \in \{1,2\}, j \in \{A, B\}.$$

Hence, the mean cycle time for each job type can be determined using *Little's theorem*:

$$CT_A = \sum_i CT_{iA} = \sum_i \frac{\bar{n}_{iA}}{TH_A}$$

$$CT_B = \sum_i CT_{iB} = \sum_i \frac{\bar{n}_{iB}}{TH_B}$$

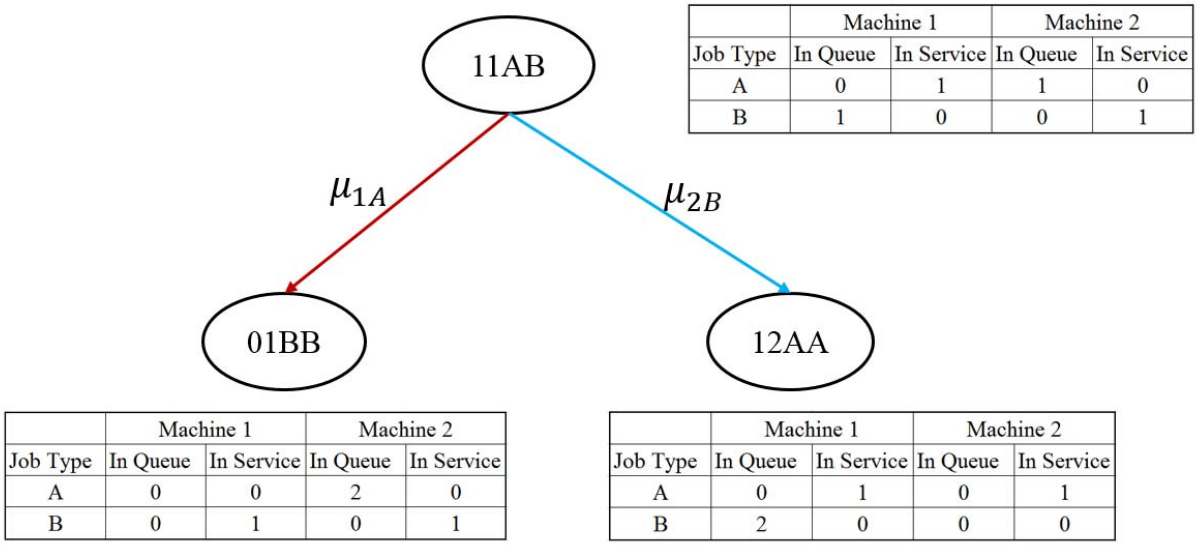
4.5.3 Non-Preemptive Priorities

The structure of the model used in this section is the same as that in last section. The only difference is that the priority scheme is non-preemptive, *i.e*., a process cannot be interrupted once started until it is finished. Although job type A has a higher priority than job type B at machine 1, it cannot "*preempt*" job type B if job type B is already in service. Because of this "non-preemptive" characteristic, the model becomes much more complicated.

As shown in last section, if the priority discipline is preemptive, it would be sufficient only to remember the number of jobs for each type at machine 1. The non-preemptive priority case is quite different. The system can be defined as the number of jobs of both types at machine 1 at an arbitrary moment in time and we denote the system representation with the tuple $(n_A, n_B, \mathsf{C}_1, \mathsf{C}_2)$. The first two parameters of this tuple, n_A and n_B $(n_A = 0,1, \dots, N_A, n_B = 0,1, \dots, N_B)$, respectively indicate the number of type A and type B jobs at machine 1, including the one that is possible in service. The indices C_1 and C_2 refer to the types of job in progress respectively at machine 1 and machine 2. Consequently, their values may be A, B or *Null* (the respective machine is empty). It is necessary to include additional information in the system representation description because the priority scheme here is non-preemptive and the service rates are unequal. The machine at which a job completes determines the types of job that will be selected for service next.

We use an example to demonstrate the meaning of the above non-preemptive system representation. First, consider the state $(1,1, A, B)$. If job type A at machine 1 finishes first, it will be replaced by a class of type B job since there is only one type B job and no type A job at machine 1. The new state entered is $(0,1, B, B)$. If job type B at machine 2 finishes first, it will be replaced by a class of type A job since there is only one type A job and two type A job at

machine 1. The new state entered is $(1,2,A,A)$. See Figure 4.7. Next, consider the state $(1,1,B,A)$, which is similar to state $(1,1,A,B)$ except for the position of the jobs in service. The next state will be state $(2,1,B,B)$ if type A job at machine 2 finishes first or state $(1,0,A,A)$ if type B job at machine 1 finishes first. Obviously, this makes a huge difference for both classes of jobs.



	Machine 1		Machine 2	
Job Type	In Queue	In Service	In Queue	In Service
A	0	1	1	0
B	1	0	0	1

	Machine 1		Machine 2	
Job Type	In Queue	In Service	In Queue	In Service
A	0	0	2	0
B	0	1	0	1

	Machine 1		Machine 2	
Job Type	In Queue	In Service	In Queue	In Service
A	0	1	0	1
B	2	0	0	0

Figure 4.7 An Example of States Flow

To find the exact analytical results for this non-preemptive model, we need to obtain the stationary probability distribution first for the number of jobs in the system. Then, the average throughput and average cycle time for each job class can be determined from these stationary probabilities.

The state space discussed here is a collection of all possible states and forms a countable set. The sequence of states the system enters over time can be described by a Markov Chain. The most common tools to study a Markov Chain are the state transition diagram and the state transition matrix. In the transition diagram, each node represents a state of the system, and the arcs indicate

permissible transitions between states. The arc labels reflect the rates at which these transitions occur. The associated state transition diagram for the two-class, two-server non-preemptive priority queueing network appears in Figure 4.8. In Figure 4.8, the states are ordered in columns according to the total number of jobs present at machine 1. The diagram depicts a system in which $N_A = 2, N_B = 2$, *i.e.*, the maximum number of jobs in the system is four. Clearly, the diagram will explode as the number of jobs increases.

Figure 4.9 reflects the corresponding one-step transition matrix for the model shown in Figure 4.8.

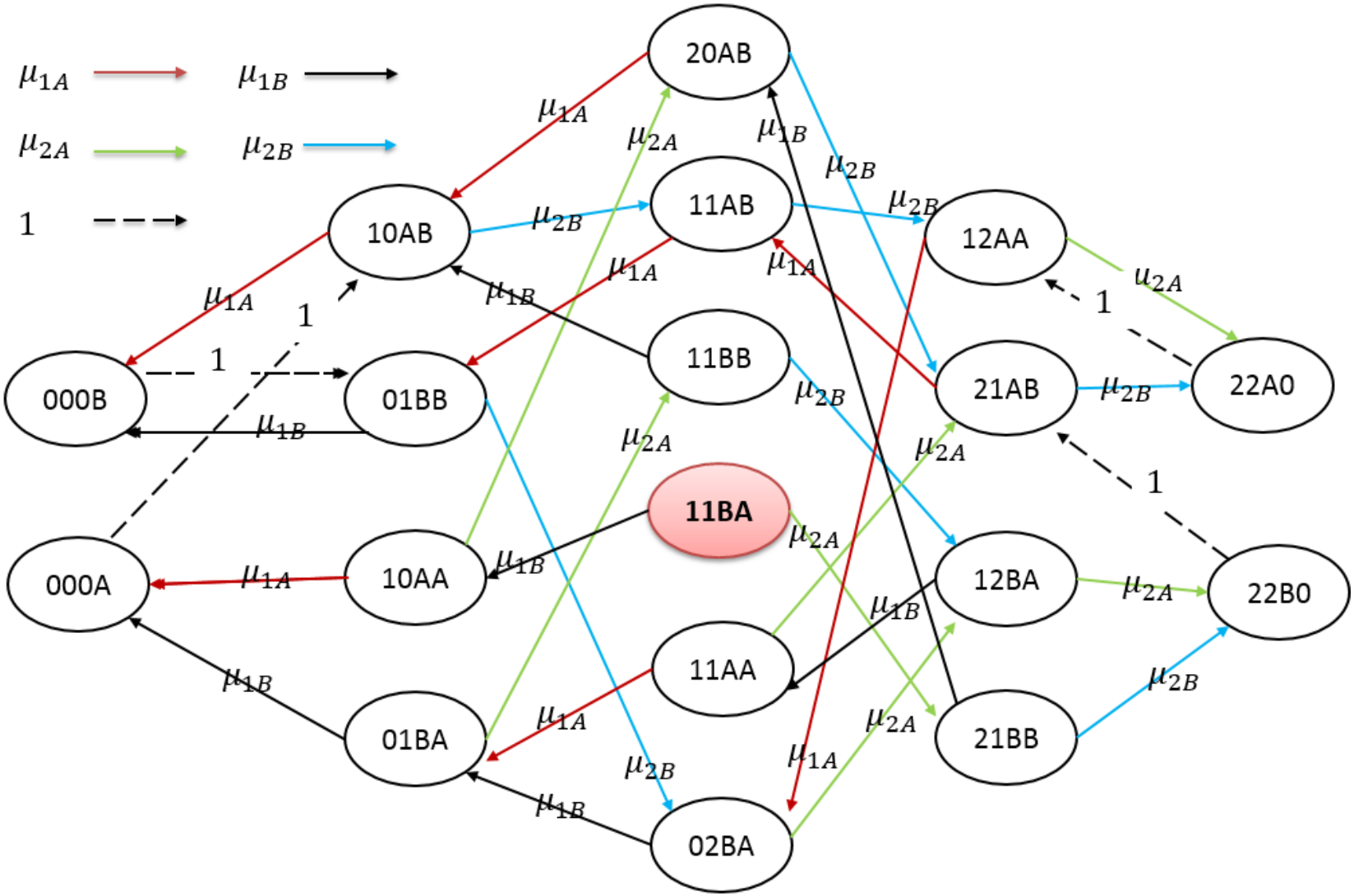


Figure 4.8 State Transition Diagram for Two-Class, Two-Server Mixed Non-Preemptive Priority System *($N_A = 2, N_B = 2$)*

□	(000A)	(000B)	(01BA)	(01BB)	(02BA)	(10AA)	(10AB)	(11AA)	(11AB)	(11BA)	(11BB)	(12AA)	(12BA)	(20AB)	(21AB)	(21BB)	(22A0)	(22B0)
(000A)	□	□	□	□	□	□	1	□	□	□	□	□	□	□	□	□	□	□
(000B)	□	□	□	1	□	□	□	□	□	□	□	□	□	□	□	□	□	□
(01BA)	$\frac{\mu_{1B}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□	□	□
(01BB)	□	$\frac{\mu_{1B}}{\mu_{2B}+\mu_{1B}}$	□	□	$\frac{\mu_{2B}}{\mu_{2B}+\mu_{1B}}$	□	□	□	□	□	□	□	□	□	□	□	□	□
(02BA)	□	□	$\frac{\mu_{1B}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□
(10AA)	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2A}}$	□	□	□	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{1A}+\mu_{2A}}$	□	□	□	□
(10AB)	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□	$\frac{\mu_{2B}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□	□	□	□
(11AA)	□	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2A}}$	□	□	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{1A}+\mu_{2A}}$	□	□	□
(11AB)	□	□	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□	□	$\frac{\mu_{2B}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□
(11BA)	□	□	□	□	□	$\frac{\mu_{1B}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{2A}+\mu_{1B}}$	□	□
(11BB)	□	□	□	□	□	□	$\frac{\mu_{1B}}{\mu_{1B}+\mu_{2B}}$	□	□	□	□	□	$\frac{\mu_{2B}}{\mu_{1B}+\mu_{2B}}$	□	□	□	□	□
(12AA)	□	□	□	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2A}}$	□	□	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{1A}+\mu_{2A}}$	□
(12BA)	□	□	□	□	□	□	□	$\frac{\mu_{1B}}{\mu_{2A}+\mu_{1B}}$	□	□	□	□	□	□	□	□	□	$\frac{\mu_{2A}}{\mu_{2A}+\mu_{1B}}$
(20AB)	□	□	□	□	□	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□	□	$\frac{\mu_{2B}}{\mu_{1A}+\mu_{2B}}$	□	□	□
(21AB)	□	□	□	□	□	□	□	□	$\frac{\mu_{1A}}{\mu_{1A}+\mu_{2B}}$	□	□	□	□	□	□	□	$\frac{\mu_{2B}}{\mu_{1A}+\mu_{2B}}$	□
(21BB)	□	□	□	□	□	□	□	□	□	□	□	□	□	$\frac{\mu_{1B}}{\mu_{2B}+\mu_{1B}}$	□	□	□	$\frac{\mu_{2B}}{\mu_{2B}+\mu_{1B}}$
(22A0)	□	□	□	□	□	□	□	□	□	□	□	1	□	□	□	□	□	□
(22B0)	□	□	□	□	□	□	□	□	□	□	□	□	□	□	1	□	□	□

Figure 4.9 One Step Transition Matrix for Two-Class, Two-Server Non-Preemptive Priority System $(N_A = 2, N_B = 2)$

The behavior of this mixed priority queueing system can be described using Continuous-Time Markov Chains (CTMCs). A CTMC is characterized by the steady-state or long-run transition rates between the states of the corresponding model. If the CTMC is ergodic, then a unique steady-state probability vector independent of the initial probability vector is given by $\boldsymbol{\pi Q} = \mathbf{0}$ where $\mathbf{Q}$ is the infinitesimal generator matrix of the CTMC. For each state of this mixed priority queueing network in equilibrium, the conservation law presents that the flux out of a state is equal to the flux into that state [Bolch, Greiner, de Meer, and Trivedi (2006)]. Thus, the flow of the steady state can be written as the following balance equations:

$$\pi_{0,0,0,B} = \pi_{1,0,A,B}\mu_{1A} + \pi_{0,1,B,B}\mu_{1B}$$

$$\pi_{0,0,0,A} = \pi_{1,0,A,A}\mu_{1A} + \pi_{0,1,B,A}\mu_{1B}$$

$$\pi_{1,0,A,B}(\mu_{1A} + \mu_{2B}) = \pi_{2,0,A,B}\mu_{1A} + \pi_{1,1,B,B}\mu_{1B}$$

$$\pi_{0,1,B,B}(\mu_{1B} + \mu_{2B}) = \pi_{0,0,0,B} + \pi_{1,1,A,B}\mu_{1A}$$

$$\pi_{1,0,A,A}(\mu_{1A} + \mu_{2A}) = \pi_{1,1,B,A}\mu_{1B}$$

$$\pi_{0,1,B,A}(\mu_{1B} + \mu_{2A}) = \pi_{1,1,A,A}\mu_{1A} + \pi_{0,2,B,A}\mu_{1B}$$

$$\pi_{2,0,A,B}(\mu_{1A} + \mu_{2B}) = \pi_{1,0,A,A}\mu_{2A} + \pi_{2,1,B,B}\mu_{1B}$$

$$\pi_{1,1,A,B}(\mu_{1A} + \mu_{2B}) = \pi_{1,0,A,B}\mu_{2B} + \pi_{2,1,A,B}\mu_{1A}$$

$$\pi_{1,1,B,B}(\mu_{1B} + \mu_{2B}) = \pi_{0,1,B,A}\mu_{2A}$$

$$\pi_{1,1,A,A}(\mu_{1A} + \mu_{2A}) = \pi_{1,2,\square,A}\mu_{1B}$$

$$\pi_{0,2,B,A}(\mu_{1B} + \mu_{2A}) = \pi_{0,1,B,B}\mu_{2B} + \pi_{1,2,A,A}\mu_{1A}$$

$$\pi_{1,2,A,A}(\mu_{1A} + \mu_{2A}) = \pi_{1,1,A,B}\mu_{2B} + \pi_{2,2,A,0}$$

$$\pi_{2,1,A,B}(\mu_{1A} + \mu_{2B}) = \pi_{2,0,A,B}\mu_{2B} + \pi_{1,1,A,A}\mu_{2A}$$

$$\pi_{1,2,B,A}(\mu_{1B} + \mu_{2A}) = \pi_{1,1,B,B}\mu_{2B} + \pi_{0,2,B,A}\mu_{2A}$$

$$\pi_{2,1,B,B}(\mu_{1B} + \mu_{2B}) = \pi_{1,1,B,A}\mu_{2A}$$

$$\pi_{2,2,A,0} = \pi_{1,2,A,A}\mu_{2A} + \pi_{2,1,A,B}\mu_{2B}$$

$$\pi_{2,2,B,0} = \pi_{1,2,B,A}\mu_{2A} + \pi_{2,1,B,B}\mu_{2B}$$

The stationary probabilities π_{n_A,n_B,C_1,C_2} are obtained by solving the above equilibrium equations, in conjunction with the normalization equation:

$$\sum_{n_A \geq 0} \sum_{n_B \geq 0} \sum_{all\ C_1,C_2} \pi_{n_A,n_B,C_1,C_2} = 1.$$

Let

$$\pi_i(\mathbf{k}) = \text{the probability that the machine } i \text{ is in state } \mathbf{S}_i = \mathbf{k},$$

In this case, $\pi_i(\mathbf{k})$ is determined by π_{n_A,n_B,C_1,C_2}.

Let $n_i = \sum_j n_{ij}$, the utilization of machine i for a class j customer is

$$\rho_{ij} = \sum_{\mathbf{k}} \pi_i(\mathbf{k}) \frac{n_{ij}}{n_i}$$

where n_{ij} is the number of class j customer at machine i.

The mean overall throughput of type A (respectively type B) jobs denoted by TH_A (respectively TH_B):

$$TH_A = \mu_{1A}\rho_{1A}$$

$$TH_B = \mu_{1B}\rho_{1B}$$

Then the mean number of jobs of j type at machine i is

$$\bar{n}_{ij} = \sum_{\mathbf{k}} n_j.\pi_i(\mathbf{k}), i \in \{1,2\}, j \in \{A, B\}.$$

Hence, the mean cycle time for each job type can be determined using *Little's theorem*:

$$CT_A = \sum_i CT_{iA} = \sum_i \frac{\bar{n}_{iA}}{TH_A}$$

$$CT_B = \sum_i CT_{iB} = \sum_i \frac{\bar{n}_{iB}}{TH_B}$$

4.6 Conditions Under Which the Non-Preemptive Discipline can be Approximated by a Preemptive Discipline

Throughout this chapter, we focus on a two-class, two-server closed queueing network. From Section 4.5, we notice that the non-preemptive priority system is much more difficult to model than the preemptive system. As the number of jobs increases, the state space explodes. It is not even possible to obtain an exact solution for large N_A and N_B. However, from Section 4.5, we also learn that it is relatively easy to obtain the exact analytical solution for the preemptive priority system. These observations motivate us to explore conditions under which the non-preemptive solution can be approximated by the preemptive solution. We tested numerous scenarios and found that under two conditions, the preemptive results approach the non-preemptive solution.

For each case, the same priority scheme as that adopted in Chapter 3 is applied. The results for preemptive and non-preemptive scenarios are compared in terms of mean overall throughput and mean overall flow time (cycle time) for specific conditions.

Condition 1: For all class j customers, if all job types have the same mean service time at the bottleneck machine, we can approximate the non-preemptive solution with preemptive solution.

Numerical Results: An example configuration (Service Time) is listed below:

Table 4.1 Example Configuration for Condition 1

	Job Type	
Machine	A	B
1	4	4
2	1	2

The mean overall throughput and mean overall flow time for preemptive and non-preemptive priority cases are compared in Figure 4.10a-4.10c. The instances are tested for $N_A + N_B = 20, 30, \text{and } 40$.

Analysis: The results confirm that the bottleneck machine is the dominant factor that impacts the system performance in terms of total mean throughput and total mean flow time. When the service times are the same for all class j customers at the bottleneck machine, the preemptive solution comes close to the non-preemptive solution.

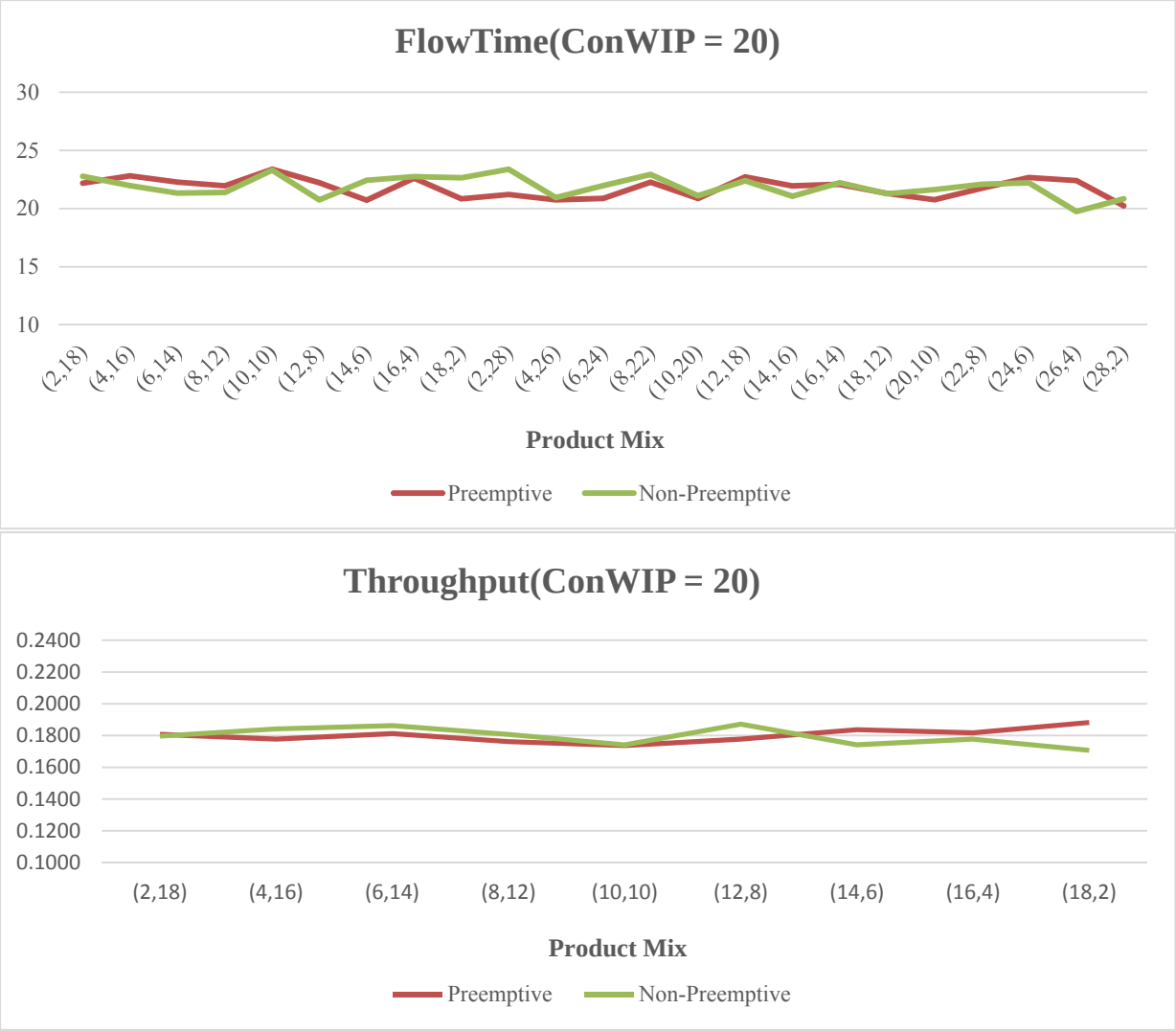


Figure 4.10a Two-Class, Two-Server Priority System ($N_A + N_B = 20$)

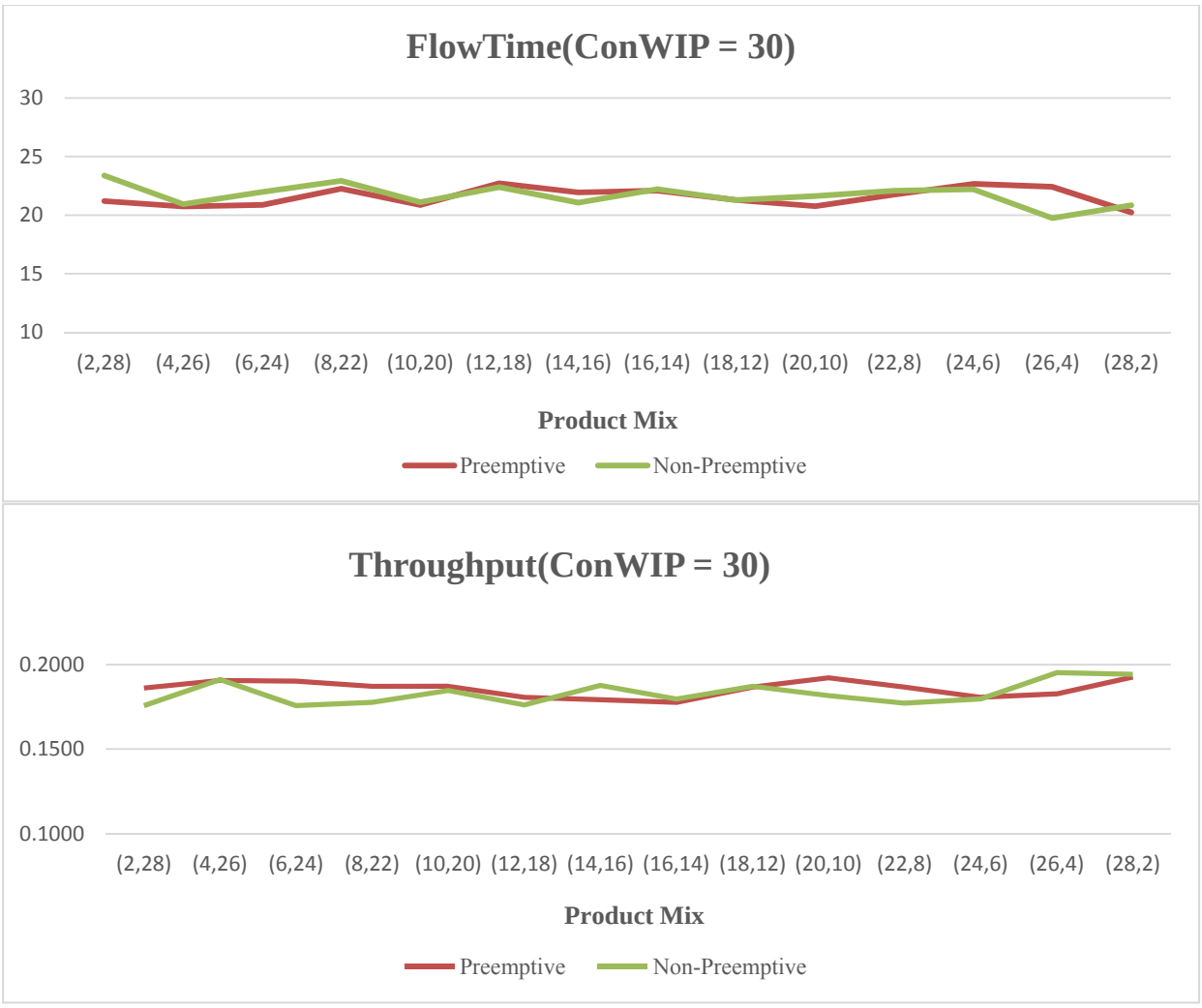


Figure 4.10b Two-Class, Two-Server Priority System ($N_A + N_B = 30$)

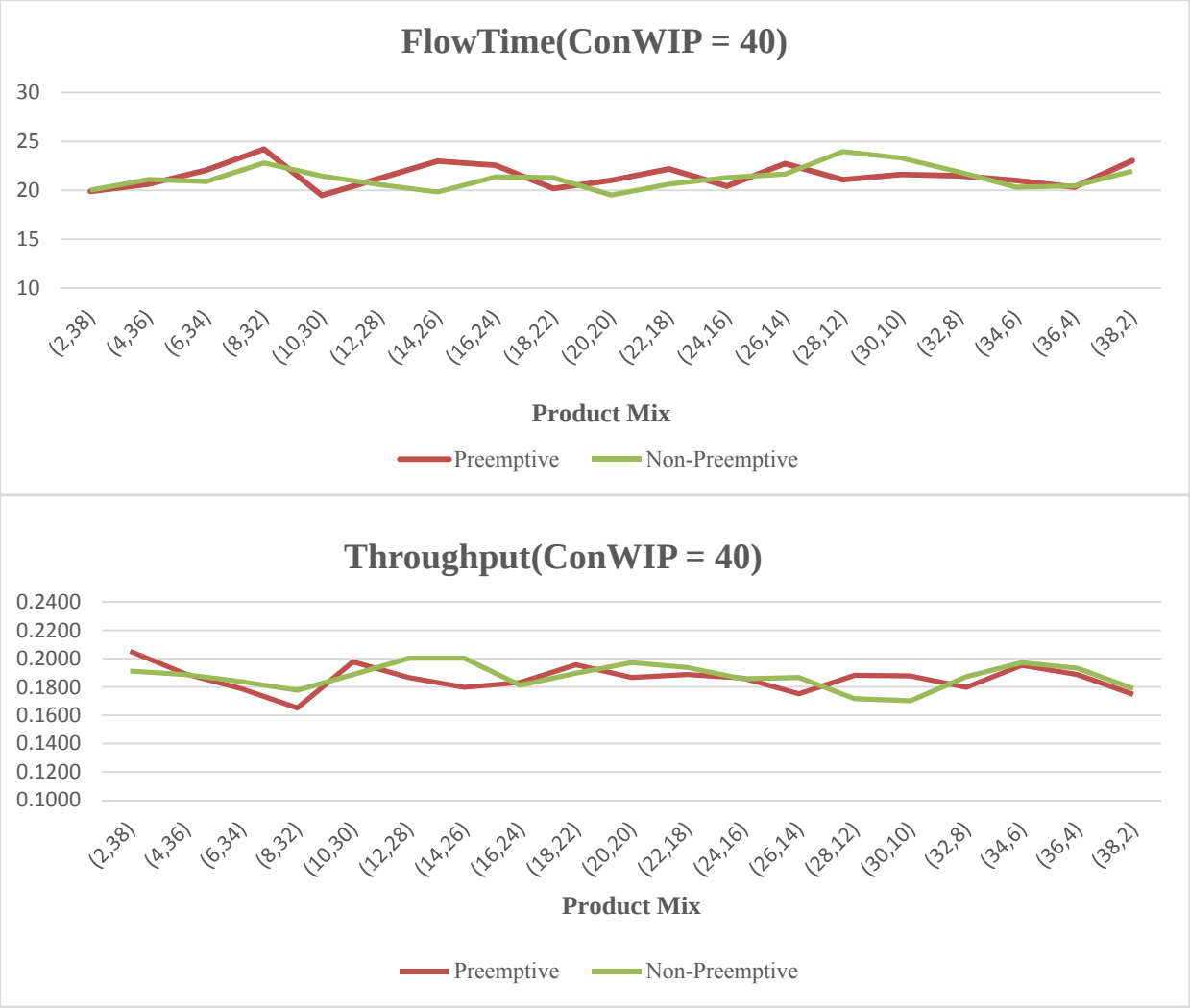


Figure 4.10c Two-Class, Two-Server Priority System ($N_A + N_B = 40$)

Condition 2: If bottleneck exists because of extremely long mean service times on bottleneck machine for all classes, we can approximate non-preemptive solutions with preemptive solutions.

Numerical Results: an example configuration (Service Time) is listed below:

Table 4.2 Example Configuration for Condition 2

	Job Type	
Machine	A	B
1	8	10
2	1	5

The mean overall throughput and mean overall flow time for preemptive and non-preemptive priority cases are compared in Figure 4.11a-4.11c. The instances are tested for $N_A + N_B = 20, 30, \text{and } 40.$

Analysis: From a modeling perspective, a lower priority job is processed as several "sliced" parts under preemptive service discipline and it is processed as a whole under non-preemptive service discipline. When the bottleneck machine is heavily loaded compared to the non-bottleneck machine, the utilization of the bottleneck machine is close to 100%. Therefore, it will make little difference between the preemptive solution and the non-preemptive solution.

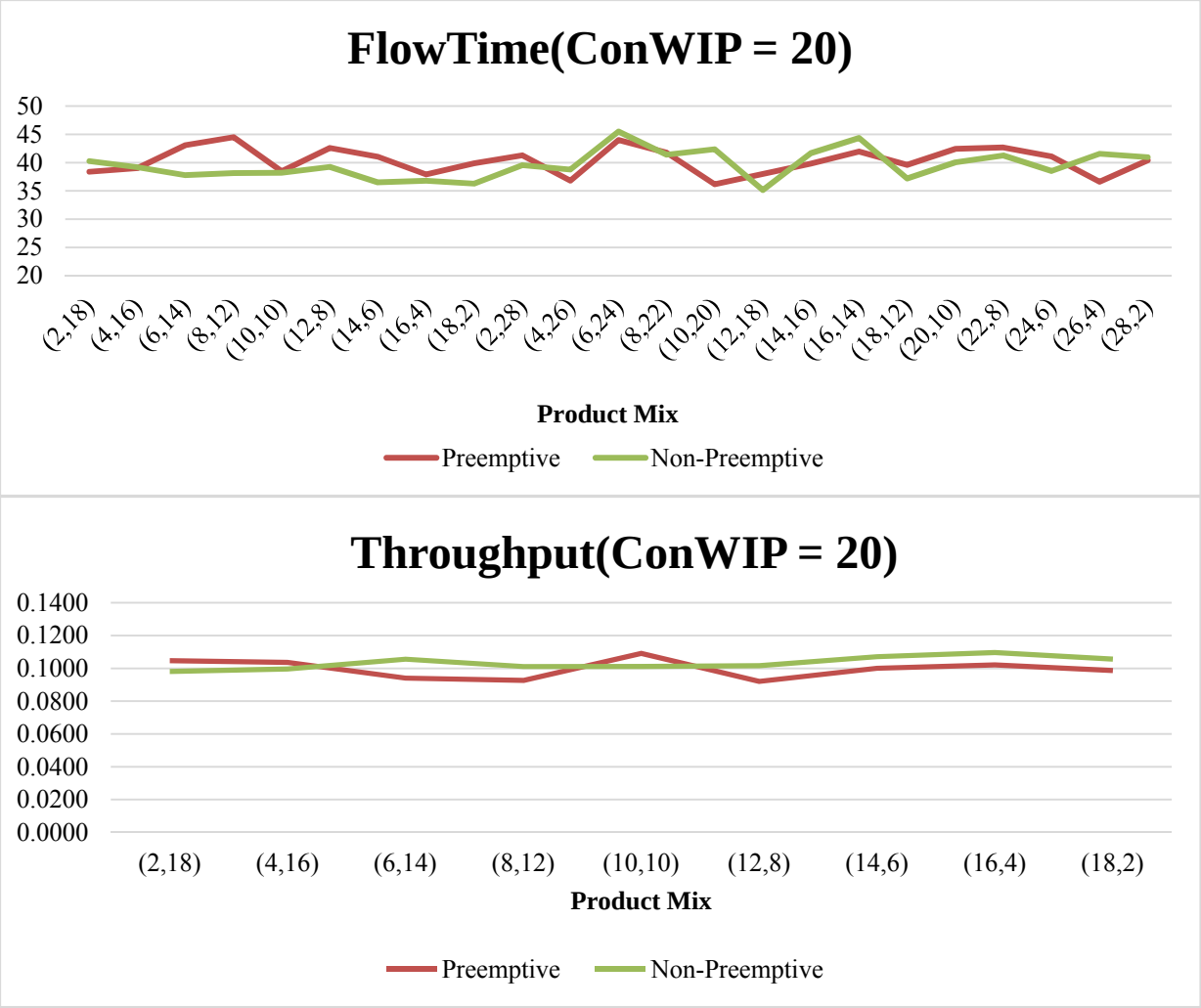


Figure 4.11a Two-Class, Two-Server Priority System ($N_A + N_B = 20$)

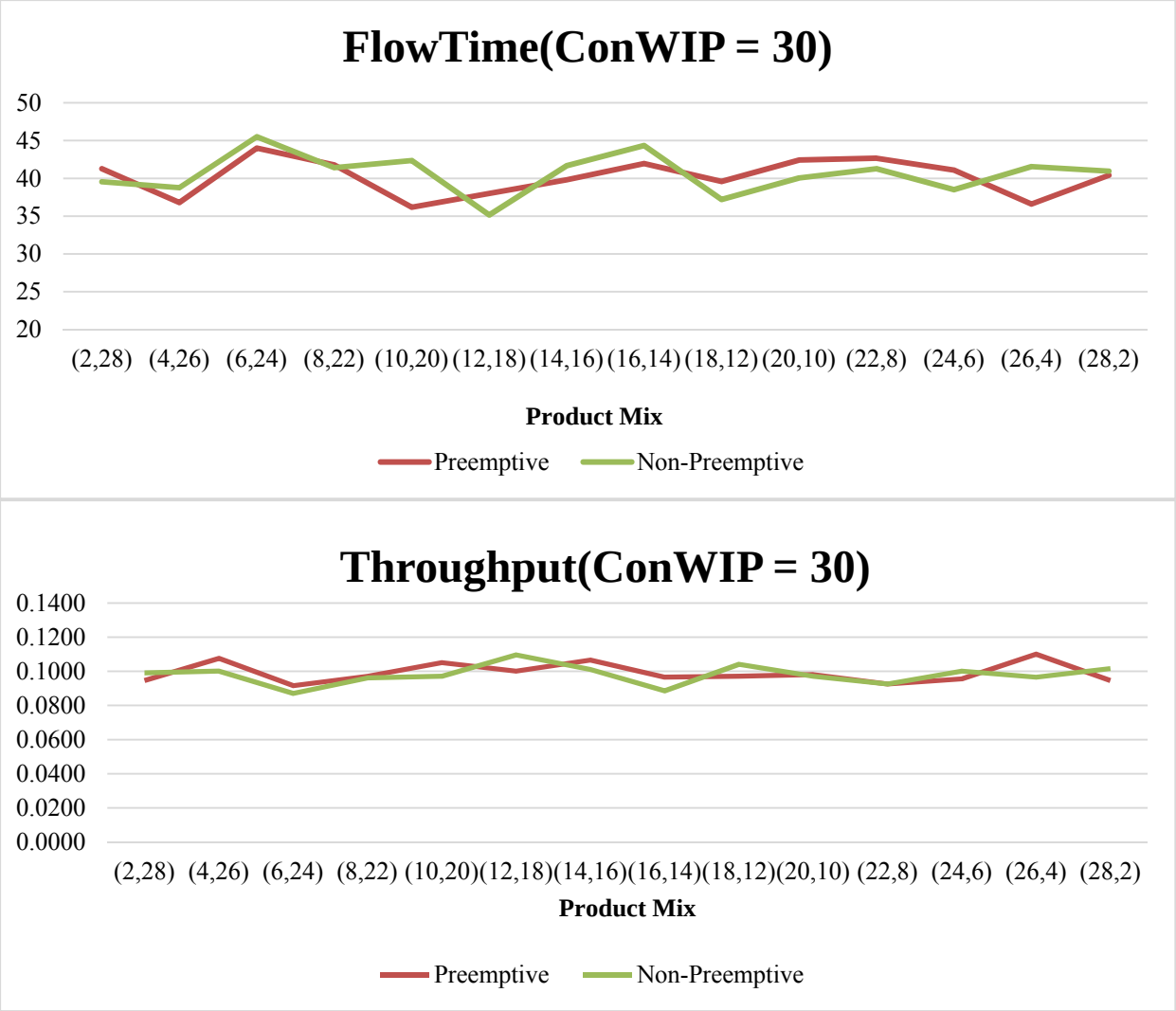


Figure 4.11b Two-Class, Two-Server Priority System ($N_A + N_B = 30$)

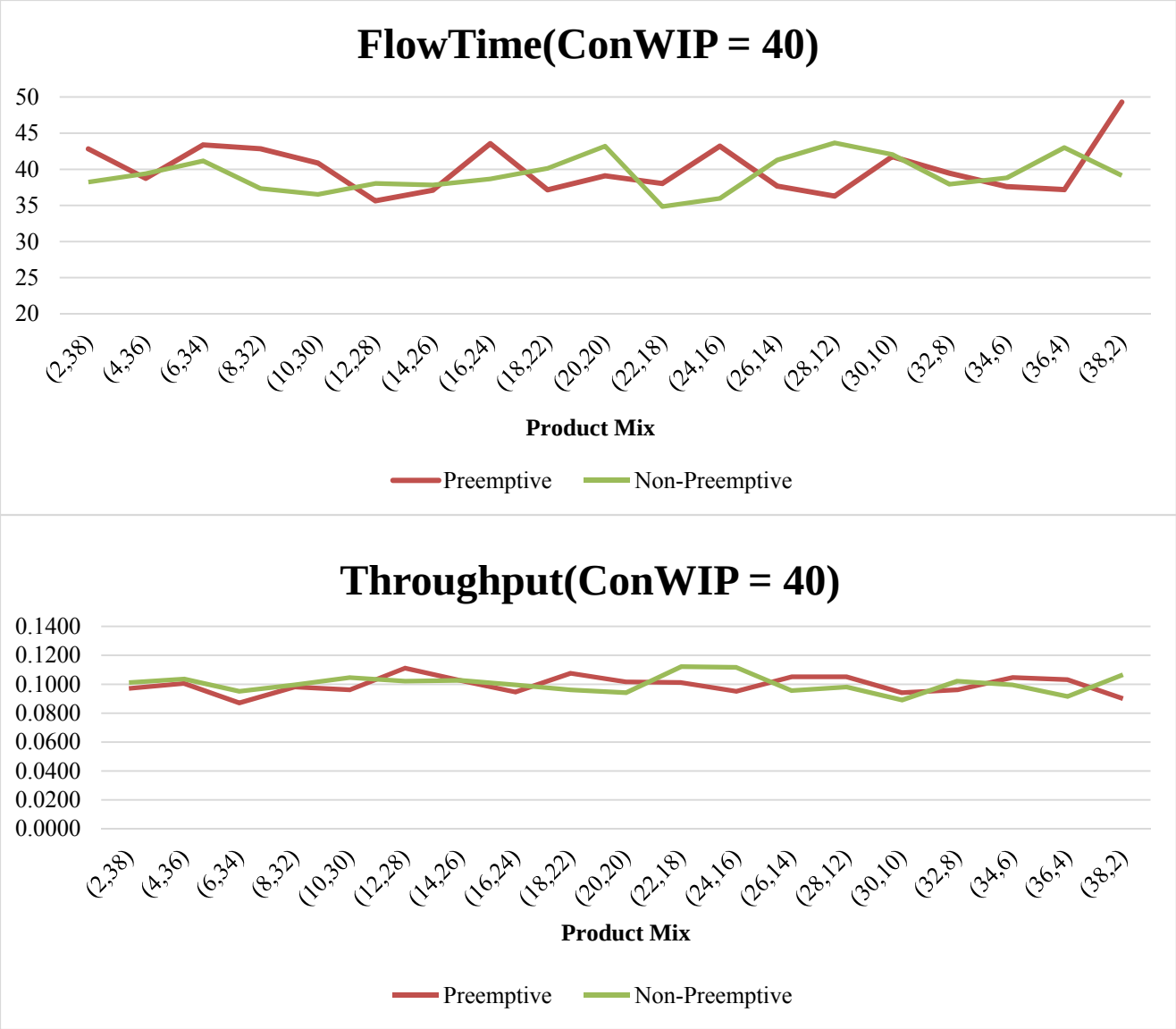


Figure 4.11c Two-Class, Two-Server Priority System ($N_A + N_B = 40$)

4.7 Summary

In this chapter, we discussed the mixed priority queueing networks, mainly a two-class, two-server mixed priority queueing system. We compared the exact solution methods for FCFS, preemptive priority, and non-preemptive priority models for $N_A = 2, N_B = 2$. Inspired by those solution methods, we found that when all job types have the same mean service time at the bottleneck machine, or when the bottleneck exists because of extremely long mean service times on the bottleneck machine for all job types, the non-preemptive priority solution can be approximated by the preemptive solution.

CHAPTER 5

CONCLUSION AND FURTHER DISCUSSIONS

5.1 Conclusion

This research study a typical MRO process which involves fixing the defective parts of airplane engines. First, the engines are disassembled into their component parts; then, these parts are tested independently, and all the impaired parts are conveyed to a back-shop for repair; finally, repaired parts are returned for reassembling the equipment. Because of the existence of resource contention (machine sharing), scheduling the back-shop for smooth flow often requires prioritizing the repair of component parts from different original assemblies at different machines. A multi-class queueing network with a ConWIP execution system is applied to model the back-shop operations. To maximize the system performance, we introduce a new priority scheme. In this scheme, we identify the bottleneck machine based on overall workload and classify machines into two categories: bottleneck machine and non-bottleneck machines. Engine part with the lowest mean cycle time receives the highest priority at the bottleneck machine and the lowest priority on non-bottleneck machines. Experimental results show that this priority scheme increases the system performance by lowering average cycle times without adversely impacting total throughput.

The job-shop scheduling problem has been studied extensively in the last a few decades. Nevertheless, only limited basic research to support their successful design and implementation has been performed. In this dissertation, we develop a new rule-based priority scheme for job-shop scheduling. We first build a simulation model that captures all the essential features of the system: ConWIP protocol, disassembly/reassembly process, non-preemptive priority scheme,

and multiple job classes. The simulation model provides valuable insights into how the new priority scheme works for various scenarios.

To our best knowledge, the priority scheme we present here is a new approach to handle job-shop scheduling. Unlike other existing models, it is robust and can be easily implemented in large scale production systems. The priority scheme is simple, efficient, and has not been analyzed before. It increases the system performance by lowering the total mean flow time without adversely impacting the total mean throughput. Although we assume that all service times are exponentially distributed, the scheme can be applied under other stochastic distributions. Furthermore, the priority scheme incorporates the idea of "balancing the flows" of the system in the sense that the utilization for both bottleneck machine and non-bottleneck machines are improved.

Furthermore, we study the exact solution methods for a two-class, two-server priority queueing system. Unlike Morris' paper and other researchers' work, we provide an exact solution method for the non-preemptive priority system in addition to the preemptive case. Also, we observed two interesting phenomena in our study:1) when all job types have the same mean service time at the bottleneck machine, or 2) when the bottleneck exists because of extremely long mean service times on the bottleneck machine for all job types, the non-preemptive priority solution can be approximated by the preemptive solution. These observations provide a new way to tackle the non-preemptive priority system when the size of the product mix is large.

5.2 Further Research Directions

The methodology can be extended in two possible ways.

1) Dynamic routings. This research work focuses on static routings, *i.e.*, all routings are fixed and pre-determined and no alternative routings or dynamic routings are allowed. The advantage of this method is easy implementation. However, this may not be the most efficient way to utilize the system resources. Dynamic routing protocols will update the routings and determine the next-best path if the regular best path to a destination becomes unusable due to machinery breakdowns or system failures. The capability to compensate for smooth flows is the most important advantage dynamic routing offers over static routing.

2) Matrix-geometric method (MGM) [Neuts (1981)]. The two-class, two-server closed priority model does not meet the requirements needed for a "classical" closed-form solution. The MGM can be applied to a multidimensional state space. It can be used to analyze continuous-time Markov chains whose transition rate matrices possess a repetitive structure like the model we discussed in this chapter.

REFERENCES

Anselmi, J. (2008). *A new framework supporting the bottleneck analysis of multiclass queueing networks*. Paper presented at the Proceedings of the 3rd International Conference on Performance Evaluation Methodologies and Tools, Athens, Greece.

Balbo, G., & Serazzi, G. (1996). Asymptotic analysis of multiclass closed queueing networks: Common bottleneck. *Performance Evaluation, 26*(1), 51-72.

Balbo, G., & Serazzi, G. (1997). Asymptotic analysis of multiclass closed queueing networks: Multiple bottlenecks. *Performance Evaluation, 30*(3), 115-152.

Baskett, F., Chandy, K. M., Muntz, R. R., & Palacios, F. G. (1975). Open, Closed, and Mixed Networks of Queues with Different Classes of Customers. *J. ACM, 22*(2), 248-260.

Bitran, G., & Dasu, S. (1992). A review of open queueing network models of manufacturing systems. *Queueing Systems, 12*(1-2), 95-133.

Bitran, G. R., & Tirupati, D. (1988). Multiproduct Queueing Networks with Deterministic Routing: Decomposition Approach and the Notion of Interference. *Management Science, 34*(1), 75-100.

Bolch, G., Greiner, S., de Meer, H., & Trivedi, K. S. (2006). *Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications*: Wiley.

Buzacott, J. A., & Shanthikumar, J. G. (1985). On Approximate Queueing Models of Dynamic Job Shops. *Management Science, 31*(7), 870-887.

Buzen, J. P. (1980). *Queueing network models of multiprogramming*: Garland Pub.

Casale, G., & Serazzi, G. (2004, 8-8 Oct. 2004). *Bottlenecks identification in multiclass queueing networks using convex polytopes.* Paper presented at the Modeling, Analysis, and Simulation of Computer and Telecommunications Systems, 2004. (MASCOTS 2004). Proceedings. The IEEE Computer Society's 12th Annual International Symposium on.

E. L. Lawler, J. K. L., A. H. G. R. Kan, D. B. Shmoys. (1993). Handbooks in OR & MS, chapter Sequencing and Scheduling: Algorithms and Complexity. *4*.

Erlang, A. K. (1917). Solution of some Problems in the Theory of Probabilities of Significance in Automatic Telephone Exchanges. *Elektrotkeknikeren, 13*.

Fan, K., & Ren-qian, Z. (2010, 9-11 July 2010). *An analysis of research in job shop scheduling problem.* Paper presented at the Advanced Management Science (ICAMS), 2010 IEEE International Conference on.

Garey, M. R. A., & Johnson, D. S. (1979). *Computers and Intractability: A Guide to the Theory of NP-Completeness*: W. H. Freeman.

Gautam, N. (2012). *Analysis of Queues: Methods and Applications (Operations Research Series)* (1 edition ed.): CRC Press.

Gere, W. S., Jr. (1966). Heuristics in Job Shop Scheduling. *Management Science, 13*(3), 167-190.

Giffler, B., & Thompson, G. L. (1960). Algorithms for Solving Production-Scheduling Problems. *Operations Research, 8*(4), 487-503.

Gold, H. (1998). A Markovian single server with upstream job and downstream demand arrival stream. *Queueing Systems, 30*(3-4), 435-455.

Goldratt, E. M., Cox, J., & Whitford, D. (2004). *The Goal: A Process of Ongoing Improvement*: Gower.

Goldratt, E. M., & Fox, R. E. (1986). *The Race*: North River Press Publishing Corporation.

Gordon, W. J., & Gordon, F. N. (1967). Closed Queuing Systems with Exponential Servers. *Operations Research, 15*(2), 254-265.

Greiner, S., Bolch, G., & Begain, K. (1998). A generalized analysis technique for queuing networks with mixed priority strategy and class switching. *Computer Communications, 21*(9), 819-832.

Gross, D., & Harris, C. M. (1974). *Fundamentals of queueing theory*: Wiley.

Haupt, R. (1989). A survey of priority rule-based scheduling. *Operations-Research-Spektrum, 11*(1), 3-16.

Hemachandra, N., & Eedupuganti, S. K. (2003). Performance analysis and buffer allocations in some open assembly systems. *Computers & Operations Research, 30*(5), 695-704.

Heyman, D. P., & Sobel, M. J. (1984). *Stochastic Models in Operations Research* (Vol. 1 and 2): McGraw-Hill

Hopp. (2008). *Factory Physics*: McGraw-Hill Irwin Irwin.

J. A. Buzacott, J. G. S. (1993). *Stochastic Models of Manufacturing Systems*: Prentice-Hall.

J. George Shanthikumar, U. S. (1988). Approximations for the time spent in a dynamic job shop with applications to due-date assignment. *International Journal of Production Research, 26*(8).

Jackson, J. R. (1954). Notes on Some Scheduling Problems. *Research Report No. 35, Management Science Research Project*.

Jackson, J. R. (1957). Networks of Waiting Lines. *Operations Research, 5*(4).

Jackson, J. R. (1963). Jobshop-like Queueing Systems. *Management Science, 10*(1), 131-142.

Jain, A., & Meeran, S. (1999). A State of the Art Review of Job-shop Scheduling Techniques, *European Journal of Operations Research*, 113.

Jaiswal, N. K. (1968). *Priority queues*: Academic Press.

Kant, K. (1992). *Introduction To Computer System Performance Evaluation*: Mcgraw-Hill College.

Kendall, D. G. (1953). Stochastic Processes Occurring in the Theory of Queues and their Analysis by the Method of the Imbedded Markov Chain. *The Annals of Mathematical Statistics, 24*(3), 338-354.

Kleinrock, L. (1965). A conservation law for a wide class of queueing disciplines. *Naval Research Logistics Quarterly, 12*(2), 181-192.

Kleinrock, L. (1975). *Queueing Systems. Volume 1: Theory* Wiley-Interscience.

Krishnamurthy, A., Suri, R., & Vernon, M. (2004). Analysis of a Fork/Join Synchronization Station with Inputs from Coxian Servers in a Closed Queuing Network. *Annals of Operations Research, 125*(1-4), 69-94.

Kühn, P. (1976). Analysis of complex queueing networks by decomposition. *Congressbook, 8th ITC*.

Lavenberg, S. S. (1980). *Closed Multichain Product Form Queueing Networks with Large Population Sizes*: International Business Machines Corporation. Research Division.

Law, A. (2006). *Simulation Modeling and Analysis with Expertfit Software* (4 edition ed.): McGraw-Hill Science/Engineering/Math.

Lazowska, E. D. (1984). *Quantitative System Performance, Computer System Analysis Using Queuing Network Models*: Prentice Hall

Maglaras, C., & Zeevi, A. (2003). Pricing and Capacity Sizing for Systems with Shared Resources: Approximate Solutions and Scaling Relations. *Management Science, 49*(8), 1018-1038.

Maglaras, C., & Zeevi, A. (2004). Diffusion Approximations for a Multiclass Markovian Service System with "Guaranteed" and "Best-Effort" Service Levels. *Mathematics of Operations Research, 29*(4), 786-813.

Maglaras, C., & Zeevi, A. (2005). Pricing and Design of Differentiated Services: Approximate Analysis and Structural Insights. *Operations Research, 53*(2), 242-262.

Marie, R. A. (1979). An Approximate Analytical Method for General Queueing Networks. *Software Engineering, IEEE Transactions on, SE-5*(5), 530-538.

Mee, R. (2009). *A Comprehensive Guide to Factorial Two-Level Experimentation* (1 edition ed.): Springer.

Morris, R. J. T. (1981). Priority Queuing Networks. *THE BELL SYSTEM TECHNICAL JOURNAL, 60*(8), 1745-1769.

Muth, J. F., & Thompson, G. L. (1963). *Industrial Scheduling*: Prentice-Hall.

Neuts, M. F. (1981). *Matrix-Geometric Solutions in Stochastic Models: An Algorithmic Approach*: Dover Publications.

Norris, J. R. (1998). *Markov Chains*: Cambridge University Press.

Paul J. Schweitzer, G. S., Marco Broglia. (1993). A Survey of Bottleneck Analysis in Closed Networks of Queues. In R. D. N. Lorenzo Donatiello (Ed.), *Performance Evaluation of Computer and Communication Systems, Joint Tutorial Papers of Performance '93 and Sigmetrics '93, Santa Clara, CA, USA, May 10-14, 1993* (Vol. 729, pp. 491-508): Springer.

Pittel, B. (1979). Closed Exponential Networks of Queues with Saturation: The Jackson-Type Stationary Distribution and Its Asymptotic Analysis. *Mathematics of Operations Research, 4*(4), 357-378.

Randhawa, R. S., & Kumar, S. (2009). Multiserver Loss Systems with Subscribers. *Math. Oper. Res., 34*(1), 142-179.

Reiser, M. (1979). A Queueing Network Analysis of Computer Communication Networks with Window Flow Control. *Communications, IEEE Transactions on, 27*(8), 1199-1209.

Reiser, M., & Kobayashi, H. (1975). Queuing networks with multiple closed chains: theory and computational algorithms. *IBM J. Res. Dev., 19*(3), 283-294.

Reiser, M., & Lavenberg, S. S. (1980). Mean-Value Analysis of Closed Multichain Queuing Networks. *J. ACM, 27*(2), 313-322.

Robertazzi, T. G. (1990). *Computer Networks and Systems: Queueing Theory and Performance Evaluation*: World Publishing Company.

Rumsewicz, M., & Henderson, W. (1989). Closed two node priority queueing networks. *European Journal of Operational Research, 38*(2), 184-201

Ryan, S., Baynat, B., & Choobineh, F. F. (2000). Determining inventory levels in a CONWIP controlled job shop. *IIE Transactions, 32*(2), 105-114.

Schmidt, J. W., & Taylor, R. E. (1970). *Simulation and analysis of industrial systems*: R. D. Irwin.

Schrage, L. (1969). Analysis and Optimization of a Queueing Model of a Real-Time Computer Control System. *IEEE Transactions on Computers, 18*(11), 997-1003.

Schrage, L. (1969). A Mixed-Priority Queue with Applications to the Analysis of Real-Time Systems. *Operations Research, 17*(4), 728-742.

Schrage, L. (1970). An Alternative Proof of a Conservation Law for the Queue G/G/1. *Operations Research, 18*(1), 185-187.

Schwetman, H. (1980). Implementing the Mean Value Algorithm for the Solution of Queueing Network Models. *Computer Science Technical Reports*.

Sisson, R. L. (1959). Methods of Sequencing in Job Shops-A Review. *Operations Research, 7*(1), 10-29.

Sisson, R. L. (1961). *Sequencing Theory, Chapter 7 in Ackoff, R. L., ed., Progress in Operations Research*: Wiley.

Sultana Parveen, H. U. (2010). Review on Job-shop and Flow-shop Scheduling. *Journal of Mechanical Engineering 41*(2).

Suri, R. (1983). Robustness of queuing network formulas. *J. ACM, 30*(3), 564-594.

Veronique Sels, N. G. a. M. V. (2012). A comparison of priority rules for the job shop scheduling problem under different flow time- and tardness-related objective functions. *International journal of production research, 50*(15).

Walrand, J. (1988). *An introduction to queueing networks*: Prentice Hall.

Whitt, W. (1983). The Queueing Network Analyzer. *Bell System Technical Journal, 62*(9).

Wolff, R. W. (1989). *Stochastic Modeling and the Theory of Queues*: Prentice Hall PTR.

VITA

Shuping Zhang is a Ph.D. candidate in the Management Science program at the University of Tennessee, Knoxville. He received a Bachelor's degree in Accounting from Shanxi University of Finance & Economics (China) in 1991, a Master's degree in Economics from Middle Tennessee State University in 2006, and a Master's degree in Statistics from the University of Tennessee, Knoxville, in 2013. He also served as a senior financial manager at China Post & Telecommunication. His current research interests include supply chain management, demand forecasting, operations research, and predictive modeling.